Addendum

Invention Title

METHOD OF OPERATING A LITHOGRAPHIC APPARATUS OR LITHOGRAPHIC PROCESSING CELL, LITHOGRAPHIC APPARATUS AND LITHOGRAPHIC PROCESSING CELL

Method of Operating a Lithographic Apparatus or Lithographic Processing Cell, Lithographic Apparatus and Lithographic Processing Cell

Field

The present invention relates to lithographic apparatus and lithographic processing cells, and the methods of operating such.

Background

A lithographic apparatus is a machine that applies a desired pattern onto a target portion of a substrate. Lithographic apparatus can be used, for example, in the manufacture of integrated circuits (ICs). In that circumstance, a patterning device, such as a mask, may be used to generate a circuit pattern corresponding to an individual layer of the IC, and this pattern can be imaged onto a target portion (e.g. comprising part of, one or several dies) on a substrate (e.g. a silicon wafer) that has a layer of radiation-sensitive material (resist). In general, a single substrate will contain a network of adjacent target portions that are successively exposed. Known lithographic apparatus include so-called steppers, in which each target portion is irradiated by exposing an entire pattern onto the target portion in one go, and so-called scanners, in which each target portion is irradiated by scanning the pattern through the projection beam in a given direction (the “scanning”-direction) while synchronously scanning the substrate parallel or anti-parallel to this direction.

In a factory, commonly referred to as a "fab" or "foundry", in which semiconductor devices are manufactured, each lithographic apparatus is commonly grouped with a "track" comprising substrate handling devices and pre- and post- processing devices to form a "lithocell". Both the lithographic apparatus and the track typically have supervisory control systems which are themselves under the control of a further supervisory control system. Substrates, which may be blank or have already been processed to include one or more process or device layers, are delivered to the lithocell in lots (also referred to as batches) for processing. A lot is, in general, a group of substrates which are to processed by the lithocell in the same way and is accompanied by a "recipe" which specifies the processes to be carried out. The lot size may be arbitrary or determined by the size of carrier used to transport substrates around the fab. The recipe may include details of the resist coating to be applied, temperature and duration of pre- and post- exposure bakes, details of the pattern to be exposed and the exposure settings for that, development duration, etc. A large number of tasks must be performed to complete the recipe for a given batch and there are many possible ways these can be done, as in many cases both the track and lithographic apparatus are capable of performing multiple tasks at once, e.g. if the track includes multiple spin coaters or multipurpose stations or if the lithographic apparatus is a dual stage apparatus having measurement and exposure stations. Thus scheduling the tasks to be performed, and optimizing that schedule, e.g. to maximize throughput, is a complex task.

In most cases, on-the-fly scheduling is limited and most sequences of tasks are hard-coded in the control software of the apparatus or the supervisory control system. A more flexible approach to scheduling is to construct a tree based on tasks to be completed and their precedence relation. In such a tree, starting from an origin, branches represent possible tasks that may be carried out and lead to leaves, from which further branches represent tasks that may then be carried out, and so on. Scheduling then becomes a matter of selecting a path through the tree. However such scheduling may not take into account the restrictions caused by the physical layout of the apparatus, nor the possibility of choices between tasks.

Summary

Accordingly, it would be advantageous, for example, to provide an improved method of scheduling tasks in a machine such as a lithographic apparatus and lithographic processing cell.

According to an aspect, there is provided a method of generating a schedule for operation of a machine forming at least a part of a lithographic apparatus or a lithographic processing cell, the method comprising:

receiving a plurality of weight factors for respective ones of a plurality of qualities affecting the outcome of a lithographic process;

generating an optimum schedule of tasks to be performed to complete said lithographic process, said optimum schedule being one whose outcome has a maximum value of total quality, where total quality is the sum of the products of the values of each of said qualities and the respective weight factors.

The method may enable improvements in machine behavior, as a run-time (optimized) schedule can be generated. Heuristics capturing effective intuitive rules for the application domain are used to direct the schedule generation process. By design-time simulation / verification, it may be ensured that the generated schedule will be valid. Optimization of the schedule, by iterating through possible/promising schedules, dynamically adapting to a better schedule as soon as one is found, while the machine continuously remains active can improve throughput and/or the quality of the manufactured devices with no additional overhead. Further optimization of the schedule by means of a post-processing step on a complete schedule, for example thus taking timing knowledge of an entire lot of substrates into account, is possible.

Optimization of a schedule may include adjustment of parameters of a task. For example an exposure or alignment task may be more accurate if performed at a slower scan speed and so optimization of the schedule may allow a slower scan speed, especially if that does not affect the critical path.

In an embodiment, the generation of schedules is carried out with reference to a model of the machine. The model of the machine basically comprises of tasks (the things to do) and resources (the things that can be employed to execute tasks). The execution order of tasks is restricted by precedence relations.

An embodiment of the present invention can be considered as an extension of generalized job shop scheduling techniques, which conventionally consider only the order of tasks and the assignment of resources. The extension comprises considering alternatives with respect to tasks, optional tasks, material constraints, resource conflicts and tied precedences.

Alternatives with respect to tasks are, in an embodiment of the invention, handled by defining groups of equivalent tasks – in a particular embodiment a group is defined as a set of nodes where a node may be a task, a cluster of tasks or another group – and a limit on the number of tasks that may be selected from the group. The number of tasks that can be selected may be a specific value, e.g. 1, indicating that exactly that number of tasks must be selected from the group, or a set of values, allowing choice over the number of tasks selected. Thus for optional tasks, the limit may be set as 0 or more. By way of an example, a substrate may be provided with 25 marker pairs of which 16 pairs must be scanned to provide a minimum level of alignment accuracy. Scanning additional marker pairs may provide additional accuracy.

Thus tasks may be defined to scan each marker and clustered in pairs. The clusters are put in a group which has a selection limit of 16 to 25.

According to another aspect of the invention, there is provided a method of operating a machine forming at least a part of a lithographic apparatus or a lithographic processing cell, the method comprising:

providing a model of the machine in an initial state;

determining eligible tasks that can be performed by the machine based on the state of said model;

selecting one or more of said tasks according to at least one predetermined criterion;

adding the one or more selected tasks to a partial schedule;

updating said model to reflect completion of said one or more selected tasks;

detecting whether the machine is idle and if so controlling it to perform said partial schedule; and

repeating said determining, selecting, adding, updating and detecting until all tasks necessary to complete a lithographic process have been scheduled.

A schedule is generated in a constructive way, starting from the beginning to determine the next “valid” or "eligible" tasks (i.e. obeying constraints of the machine and the manufacturing process) to perform. Instead of determining all possible schedules, heuristic filters can be used to direct the schedule generation. Examples of heuristic rules are First Start First, Preferred Resources, Priority Tasks.

The dispatch of partial schedules in the case that the machine is idle, eliminates any scheduling overhead. Such partial schedules may consist of only one task so also providing the benefits of state-based control.

The durations of tasks are taken into account, including the duration needed for resource preparation or setup, which may depend on the previously executed task(s) and the state in which the resource is left. Durations can be determined by several means: 1) by design, or calibration – for tasks which will always have the same duration in different contexts, the timing of each task can be measured once, and stored in the system; 2) by using dedicated mathematical functions – for tasks which depend on many parameters, which are only known during runtime, e.g. coordinate, dose, speed, acceleration, etc.; and/or 3) by dynamically monitoring durations, and using moving average (MA) values – for tasks of which the duration may change slowly over time.

The default heuristic to schedule a task may be “As Soon As Possible” (ASAP). If the machine is idle, the first task chosen will be dispatched immediately, which can be done as the schedule is generated in a constructive way. In this approach, (possibly) sub-optimal schedule solutions are permitted in favor of just getting the machine to start work. If the system is not idle, the next tasks will be added to a queue, comparable to a “heap of pieces” and a post processing step may be performed on the resulting schedule, to optimize it.

The post-processing step may involve adjusting the relative timings of tasks and adjustable parameters of tasks but not the tasks selected, the order of tasks in the schedule or resource allocations. In particular, a transport task, such as the transfer of a substrate from a substrate table after exposure or from a pre-aligner to a substrate table for exposure, may be scheduled to be carried out "As Late As Possible" – that is as late as the task can be carried out without delaying the critical path. Determination of when as late as possible is may take account of other tasks using the resource and other tasks that may interfere with the task being delayed. This approach may be beneficial where the transport task involves moving the substrate from a resource where it is conditioned, e.g. temperature controlled or bathed in clean air, since it ensures that the minimum time is spent in unconditioned surroundings.

In many cases, multiple solutions are possible for “valid / eligible" tasks. While the system is already working on a (possibly sub-optimal) schedule, the scheduler may iterate through the schedule generating process, each time selecting different valid paths, in an attempt to find a better schedule. Possibilities are explored from the beginning to the end, for the same reason as the constructive schedule generation. Each time a better schedule is found, the scheduler may change to this better schedule (cache), taking into account the tasks that have already been executed or dispatched.

To ensure that workable schedules are created, various configurable constraints may be applied. These may relate to logistic integrity, material flow deadlocks and hardware interference.

For example, to preserve logistic integrity, the scheduler may ensure that the resources assigned to execute certain tasks in the 'life of a material' are consistent: e.g. if a substrate is loaded onto a first substrate table in a dual stage apparatus, it must also be processed on that substrate table thereafter. Also, the scheduler may ensure that the combination of resources involved in material transport is feasible. For example, a mask may only be transported from a transfer robot onto one turret elevator, not onto the other. A further example is to ensure that the material capacities of the resources of the machine, either individually or in sets, are not exceeded. For example, a mask library may only be able to contain a limited number of masks, a mask table may only have one, etc.. To satisfy these additional constraints, logistic task information is defined, and logistic bookkeeping is integrated in the scheduler. Throughout construction of the schedule, the constraints are checked.

How deadlocks in a material's life may be dealt with can be illustrated by an example: if a mask is to be preloaded from a mask input lock to a turret by two subsequent tasks lock-robot, robot-turret, it must be possible to not only execute lock-robot, but also robot-turret subsequently. To avoid this type of deadlock, 'tied precedences' can be used. The decision to schedule one of the tasks that are 'tied together' considers the eligibility of the entire 'tie'. In other words, the first task is only scheduled if the entire tie can be scheduled. An example of a deadlock concerning multiple material lives is the limit that the turret and the mask table may contain only two masks in total. This means that there can be no preload if there is already a mask on the mask table and another on the turret. To avoid this type of deadlocks, a Work In Process constraint is employed.

Hardware interference constraints include that some resources have to move synchronously for certain moves: e.g. substrate table swap. These situations can be defined and taken into account by the scheduler when determining 'setup' or 'preparation' tasks. Another constraint is that some resources can collide in certain collision-hazardous areas. To avoid this type of interference, each collision-hazardous area can be defined as a mutually exclusive resource. These resources are then automatically assigned by the scheduler if the physical resource (e.g. robot) crosses the area.

The machine may be the whole of a lithographic processing unit, comprising a lithographic apparatus and a track unit comprising substrate handling devices and pre- and post- processing devices, or just the lithographic apparatus or just the track unit or just a subsystem within the lithographic apparatus or track unit.

According to a further aspect of the invention, there is provided a supervisory control system to operate a machine forming at least a part of a lithographic apparatus or a lithographic processing cell, the control system comprising:

an input device configured to receive a plurality of weight factors for respective ones of a plurality of qualities affecting the outcome of a lithographic process;

a scheduler configured to generate an optimum schedule of tasks to be performed to complete said lithographic process, said optimum schedule being one whose outcome has a maximum value of total quality, where total quality is the sum of the products of the values of each of said qualities and the respective weight factors.

According to further aspect, there are provided a lithographic processing cell, a lithographic apparatus and a track unit, each including a control system as described above.

Although specific reference may be made in this text to the use of lithographic apparatus in the manufacture of ICs, it should be understood that the lithographic apparatus described herein may have other applications, such as the manufacture of integrated optical systems, guidance and detection patterns for magnetic domain memories, liquid-crystal displays (LCDs), thin-film magnetic heads, etc. The skilled artisan will appreciate that, in the context of such alternative applications, any use of the terms "wafer" or "die" herein may be considered as synonymous with the more general terms "substrate" or "target portion", respectively. The substrate referred to herein may be processed, before or after exposure, in for example a track (a tool that typically applies a layer of resist to a substrate and develops the exposed resist) or a metrology or inspection tool. Where applicable, the disclosure herein may be applied to such and other substrate processing tools. Further, the substrate may be processed more than once, for example in order to create a multi-layer IC, so that the term substrate used herein may also refer to a substrate that already contains multiple processed layers.

The terms "radiation" and "beam" used herein encompass all types of electromagnetic radiation, including ultraviolet (UV) radiation (e.g. having a wavelength of 365, 248, 193, 157 or 126 nm) and extreme ultra-violet (EUV) radiation (e.g. having a wavelength in the range of 5-20 nm), as well as particle beams, such as ion beams or electron beams.

The term "patterning device" used herein should be broadly interpreted as referring to any device that can be used to impart a projection beam with a pattern in its cross-section such as to create a pattern in a target portion of the substrate. It should be noted that the pattern imparted to the projection beam may not exactly correspond to the desired pattern in the target portion of the substrate. Generally, the pattern imparted to the projection beam will correspond to a particular functional layer in a device being created in the target portion, such as an integrated circuit.

A patterning device may be transmissive or reflective. Examples of patterning devices include masks, programmable mirror arrays, and programmable LCD panels. Masks are well known in lithography, and include mask types such as binary, alternating phase-shift, and attenuated phase-shift, as well as various hybrid mask types. An example of a programmable mirror array employs a matrix arrangement of small mirrors, each of which can be individually tilted so as to reflect an incoming radiation beam in different directions; in this manner, the reflected beam is patterned. In each example of a patterning device, the support structure may be a frame or table, for example, which may be fixed or movable as required and which may ensure that the patterning device is at a desired position, for example with respect to the projection system. Any use of the terms "reticle" or "mask" herein may be considered synonymous with the more general term "patterning device".

The term "projection system" used herein should be broadly interpreted as encompassing various types of projection system, including refractive optical systems, reflective optical systems, and catadioptric optical systems, as appropriate for example for the exposure radiation being used, or for other factors such as the use of an immersion fluid or the use of a vacuum. Any use of the term "lens" herein may be considered as synonymous with the more general term "projection system".

The illumination system may also encompass various types of optical components, including refractive, reflective, and catadioptric optical components for directing, shaping, or controlling the projection beam of radiation, and such components may also be referred to below, collectively or singularly, as a "lens".

The lithographic apparatus may be of a type having two (dual stage) or more substrate tables (and/or two or more mask tables). In such "multiple stage" machines the additional tables may be used in parallel, or preparatory steps may be carried out on one or more tables while one or more other tables are being used for exposure.

The lithographic apparatus may also be of a type wherein the substrate is immersed in a liquid having a relatively high refractive index, e.g. water, so as to fill a space between the final element of the projection system and the substrate. Immersion liquids may also be applied to other spaces in the lithographic apparatus, for example, between the mask and the first element of the projection system. Immersion techniques are well known in the art for increasing the numerical aperture of projection systems.

Brief Description of the Drawings

Embodiments of the invention will now be described, by way of example only, with reference to the accompanying schematic drawings in which corresponding reference symbols indicate corresponding parts, and in which:

Figure 1 depicts a lithographic apparatus according to an embodiment of the invention;

Figure 2 depicts a lithographic processing cell according to an embodiment of the invention;

Figure 3 depicts a control system for the lithographic apparatus of Figure 1;

Figure 4 depicts definition levels of Task Resource Systems (TRS);

Figures 4a and b are UML class diagrams of TRS definition levels 1 and 2;

Figure 5 is a flow diagram of a method according to an embodiment of the invention;

Figure 6 depicts the mask handling arrangements in an apparatus according to an embodiment of the invention and certain tasks that may be performed by the apparatus;

Figure 7 is a Gantt chart of an example task schedule generated by a method according to an embodiment of the invention;

Figure 8 is a Gantt chart showing the critical path of the example schedule of Figure 7;

Figure 9 is a Gantt chart of a second example task schedule generated by a method according to an embodiment of the invention;

Figure 10 is a Gantt chart showing the critical path of the example schedule of Figure 9;

Figure 11 depicts the substrate handling arrangements in an apparatus according to an embodiment of the invention showing an area of possible resource conflict;

Figures 12 to 16 depict various resource automata;

Figure 17 depicts a sequence of tasks in the life of a substrate;

Figure 18 depicts a sequence of tasks in the life of a substrate omitting setup tasks;

Figure 19 depicts transport tasks and material occupation tasks;

Figure 20 is a view similar to Figure 11 also illustrating a possible deadlock;

Figure 21 is a Gantt chart of a third example task schedule generated by a method according to an embodiment of the invention;

Figure 22 is a Gantt chart showing the critical path of the example schedule of Figure 21;

Figure 23 depicts substrate handling arrangements according to a particular embodiment of the invention;

Figures 24 to 27 are automata of the resources of the particular embodiment of the invention;

Figure 28 depicts a sequence of tasks in the lives of three substrates;

Figures 29 to 32 are diagrams of the substrate handling arrangements of the particular embodiment showing possible deadlock situations.

Figure 33 is a Gantt chart of a fourth example task schedule generated by a method according to an embodiment of the invention;

Figure 34 is a Gantt chart showing the critical path of the example schedule of Figure 33;

Figure 35 is a Gantt chart of a fifth example task schedule generated by a method according to an embodiment of the invention;

Figure 36 is a Gantt chart of a sixth example task schedule generated by a method according to an embodiment of the invention; and

Figure 37 is a graph showing post exposure bake times for the schedules of Figures 33, 35 and 36.

Detailed Description

Figure 1 schematically depicts a lithographic apparatus according to a particular embodiment of the invention. The apparatus comprises:

- an illumination system (illuminator) IL for providing a projection beam PB of radiation (e.g. UV radiation or DUV radiation).
- a first support structure (e.g. a mask table) MT for supporting a patterning device (e.g. a mask) MA and connected to a first positioning device PM for accurately positioning the patterning device with respect to item PL;
- a substrate table (e.g. a wafer table) WT for holding a substrate (e.g. a resist-coated wafer) W and connected to a second positioning device PW for accurately positioning the substrate with respect to item PL; and
- a projection system (e.g. a refractive projection lens) PL for imaging a pattern imparted to the projection beam PB by patterning device MA onto a target portion C (e.g. comprising one or more dies) of the substrate W.

As here depicted, the apparatus is of a transmissive type (e.g. employing a transmissive mask). Alternatively, the apparatus may be of a reflective type (e.g. employing a programmable mirror array of a type as referred to above).

The illuminator IL receives a beam of radiation from a radiation source SO. The source and the lithographic apparatus may be separate entities, for example when the source is an excimer laser. In such cases, the source is not considered to form part of the lithographic apparatus and the radiation beam is passed from the source SO to the illuminator IL with the aid of a beam delivery system BD comprising for example suitable directing mirrors and/or a beam expander. In other cases the source may be integral part of the apparatus, for example when the source is a mercury lamp. The source SO and the illuminator IL, together with the beam delivery system BD if required, may be referred to as a radiation system.

The illuminator IL may comprise an adjusting device AM for adjusting the angular intensity distribution of the beam. Generally, at least the outer and/or inner radial extent (commonly referred to as σ-outer and σ-inner, respectively) of the intensity distribution in a pupil plane of the illuminator can be adjusted. In addition, the illuminator IL generally comprises various other components, such as an integrator IN and a condenser CO. The illuminator provides a conditioned beam of radiation, referred to as the projection beam PB, having a desired uniformity and intensity distribution in its cross-section.

The projection beam PB is incident on the mask MA, which is held on the mask table MT. Having traversed the mask MA, the projection beam PB passes through the lens PL, which focuses the beam onto a target portion C of the substrate W. With the aid of the second positioning device PW and position sensor IF (e.g. an interferometric device), the substrate table WT can be moved accurately, e.g. so as to position different target portions C in the path of the beam PB. Similarly, the first positioning device PM and another position sensor (which is not explicitly depicted in Figure 1) can be used to accurately position the mask MA with respect to the path of the beam PB, e.g. after mechanical retrieval from a mask library, or during a scan. In general, movement of the object tables MT and WT will be realized with the aid of a long-stroke module (coarse positioning) and a short-stroke module (fine positioning), which form part of the positioning devices PM and PW. However, in the case of a stepper (as opposed to a scanner) the mask table MT may be connected to a short stroke actuator only, or may be fixed. Mask MA and substrate W may be aligned using mask alignment marks M1, M2 and substrate alignment marks P1, P2.

The depicted apparatus can be used in the following preferred modes:

1. In step mode, the mask table MT and the substrate table WT are kept essentially stationary, while an entire pattern imparted to the projection beam is projected onto a target portion C in one go (i.e. a single static exposure). The substrate table WT is then shifted in the X and/or Y direction so that a different target portion C can be exposed. In step mode, the maximum size of the exposure field limits the size of the target portion C imaged in a single static exposure.

2. In scan mode, the mask table MT and the substrate table WT are scanned synchronously while a pattern imparted to the projection beam is projected onto a target portion C (i.e. a single dynamic exposure). The velocity and direction of the substrate table WT relative to the mask table MT is determined by the (de-)magnification and image reversal characteristics of the projection system PL. In scan mode, the maximum size of the exposure field limits the width (in the non-scanning direction) of the target portion in a single dynamic exposure, whereas the length of the scanning motion determines the height (in the scanning direction) of the target portion.

3. In another mode, the mask table MT is kept essentially stationary holding a programmable patterning device, and the substrate table WT is moved or scanned while a pattern imparted to the projection beam is projected onto a target portion C. In this mode, generally a pulsed radiation source is employed and the programmable patterning device is updated as required after each movement of the substrate table WT or in between successive radiation pulses during a scan. This mode of operation can be readily applied to maskless lithography that utilizes a programmable patterning device, such as a programmable mirror array of a type as referred to above.

Combinations and/or variations on the above described modes of use or entirely different modes of use may also be employed.

The lithographic apparatus LA shown in Figure 1, forms part of the lithographic processing cell, or lithocell, LO shown in Figure 2. As well as the lithographic apparatus LA, the lithocell LO includes input/output ports I/O1 and I/O2 (a single port or more than two may also be provided), chiller plates CH for cooling substrates, bake plates BK for heating substrates, spin coaters SC (typically four) for coating substrates, e.g. with resist, developers DE (again typically four) and a substrate handler, or robot, RO for moving substrates between the various devices and the loading bay LB of the lithographic apparatus LA. The aforementioned devices are generally referred to collectively as the track and are controlled by a track control unit TCU so as to process substrates according to the appropriate recipe. Typically, substrates are taken in at one of the ports I/O1 or I/O2, cooled on a chiller plate CH, coated with resist using a spin coater SC, given a pre-exposure bake on a bake plate BK to drive off excess solvent in the resist and cooled again before being exposed by the lithographic apparatus LA. After exposure, the substrates are subjected to a soft bake, cooled, developed, hard baked and cooled again before being output via one of the ports.

Co-ordination of the mechatronic systems of the lithographic apparatus LA is the responsibility of supervisory machine control SMC, which is shown in Figure 3. The supervisory machine control includes a model Mo of the machine (all or part of the lithographic apparatus), an input/output device I/O (e.g. keyboard & screen, removable disk drive or network connection) through which job parameters and other information can be entered, a schedule generator SG (described further below) and a schedule evaluator and optimizer SE. In order to optimize performance, it is desirable that supervisory control is flexible, and that it is able to evaluate 'what-if' scenarios before actual execution. As a basis for this evaluation, a model of the system is required. A scheduler, comprising schedule generator SG and evaluator SE, which is embedded in SMC takes care of the evaluation. Below are described a basis of the scheduling problem, a model of the machine and the manufacturing work to be done, and a basic algorithm of an embedded scheduler according to an embodiment of the invention.

From the supervisory machine control point of view, the machine can be considered as a task resource system (TRS). A manufacturing process can be associated with a task, whereas a mechatronic system can be associated with a resource. Optimization of machine behavior can start from several TRS definition levels, as has been described in N.J.M. van den Nieuwelaar, J.M. van de Mortel-Fronczak, J.E. Rooda, "Design of Supervisory Machine Control", European Control Conference 2003, which document is hereby incorporated in its entirety by reference. The higher the definition level, the more room there is for choices. By making choices, TRS definitions can be transformed to lower levels, to finally result in temporal machine behavior (TRS definition level 0: timed TRS, see Figure 4). In the paper "Design of Supervisory Machine Control", the lower two TRS definition levels (1, 0) and the applicable transformation functionality between them (layer A) are formally described, and the higher definition levels and the issues involved are introduced. Furthermore, some considerations to be taken into account when deciding to either make choices design-time or run-time (by SMC) are given. Finally, an overview of known techniques to support the design of SMC is discussed.

The following description of an embodiment of the invention has two parts. First, the definition level of the starting point of the optimization problem is raised from 1 to 2: an unselected TRS. Second, a solution for the optimization problem is presented that starts from a system definition of class 2, taking into account the technique considerations described in "Design of Supervisory Machine Control". Important requirements for the approach are extendibility towards definition level 3, and run-time usability in SMC. This approach forms a basis for model-based supervisory control of manufacturing machines according to an embodiment of the invention, which has several advantages compared to the common used state-based control. Using the model, supervisory control can evaluate 'what-if' scenarios with respect to control decisions, and schedule tasks in time rather than just one after another.

Moreover, as the model enables supervisory control to 'look ahead', it is possible to use this predictive information to synchronize with related (parts of) machines that are in another control scope. Furthermore, the approach is flexible, which improves its maintainability.

The optimization problem starting from an instantiated, unselected TRS definition $(\mathcal{D}_2)$ is formally defined below. Let $f_{AB}(\mathcal{D}_2) = \mathcal{D}_0$ be the optimization function that performs transformations A and B on $\mathcal{D}_2$ to return the temporal machine behavior $\mathcal{D}_0$. The constraints that have to be satisfied for $f_{AB}(\mathcal{D}_2)$ are constructed by combining the constraints of the separate transformations A and B. First, the definitions $\mathcal{D}_0$ and $\mathcal{D}_1$ and the constraints concerning transformation $A(f_A(\mathcal{D}_1))$ are summarized, which is an extraction from "Design of Supervisory Machine Control". Subsequently, the alternatives that are relevant in the application domain of manufacturing machines are analyzed, and based on this analysis $\mathcal{D}_2$ is formulated. After that, the constraints to be taken into account when selecting from the alternatives defined in definition level 2 to reach definition level 1 (transformation B or $f_B(\mathcal{D}_2)$) are discussed. Finally, the entire problem $f_{AB}(\mathcal{D}_2)$ is defined using the fact that $f_{AB}(\mathcal{D}_2) = f_A(f_B(\mathcal{D}_2))$.

The timing behavior of timed TRS $\mathcal{D}_0$ can be described by a 5-tuple $(\mathcal{T}_0, \mathcal{R}, I_0, \tau_{S_0}, \tau_{F_0})$:

$\mathcal{T}_0$ is a finite set whose elements are called tasks.

$\mathcal{R}$ is a finite set whose elements are called resources.

$I_0 : \mathcal{T}_0 \rightarrow \mathcal{P}(\mathcal{R})$ is the set of resources that are involved in a certain task.

$\tau_{S_0}, \tau_{F_0} : \mathcal{T}_0 \rightarrow \mathbb{R}^+$ are the start time and the end time of a certain task, which implies that these are the same for all involved resources.

Note that, by convention, the definition level is added to each definition element in subscript. Definition elements which are not level specific have no suffix. Furthermore, note that $\mathcal{D}_0$ can be visualized as a Gantt chart.

The constraint that has to be satisfied for the instances of the definition elements is as follows:

0a: Per resource, there is a chronological sequence of pairs of task start and task finish times.

The timing behavior of a task resource system is induced by physical state transitions of resources. Machine resources are mechatronic systems that have certain performance limitations. Therefore resources have to obey certain behavioral restrictions when executing physical state transitions. Tasks impose start and end states of these physical state transitions, as well as behavioral restrictions of the transition trajectories in between. Furthermore, tasks have to be executed in a certain order. A selected, untimed TRS $\mathcal{D}_1$ can be described by a 6-tuple $(\mathcal{T}_1, \mathcal{R}, I_1, P_l, Sb_l Se_l)$:

$\mathcal{T}_1$ and I_1 are equal to $\mathcal{T}_0$ and I_0, respectively.

$P_1 \in \mathcal{P}(\mathcal{T}_1 \times \mathcal{T}_1)$ is the precedence relation between tasks.

$Sb_l, Se_l : \mathcal{T}_1 \times \mathcal{R} \to \mathcal{S}$ give the begin and the end (physical) states of each resource involved in a certain task, where $\mathcal{S}$ is the set of all possible physical resource states.

Constraints that have to be satisfied for the instances of the definition elements are as follows:

1a: The sequence of tasks per resource is a chain (matches 0a).

1b: P_1 contains no cycles.

1c: For every resource involved in a task, the begin and end state must be defined.

These definition elements are depicted as a UML class diagram in Figure 4a.

Transformation A involves timing of the tasks of the untimed, selected TRS definition $\mathcal{D}_1$. Constraints that have to be satisfied for $f_A(\mathcal{D}_1)$ are as follows:

a1: Nothing changes with respect to tasks and the (involved) resources:

$\mathcal{T}_0 = \mathcal{T}_1, I_0 = I_1$

a2: By convention, time starts at 0. Furthermore, the finish time of a task equals its start time plus its duration:

$\forall t : t \in \mathcal{T}_1 : \tau_{S_0}(t) \geq 0 \wedge \tau_{F_0}(t) = \tau_{S_0}(t) + \tau_{t_0}(t)$.

a3: For subsequent tasks it holds that the start time of the succeeding task is greater than or equal to the finish time of the preceding task:

$\forall t, t' : (t, t') \in P_1 : \tau_{S_0}(t') \geq \tau_{F_0}(t)$.

a4: To match the states of subsequent tasks on the same resource, setup state transitions of the resource might be necessary. In these cases the start time of the succeeding task is greater than or equal to the finish time of the preceding task plus the duration of the setup resource state transition between the tasks:

$$\forall t,t',r : (t,t') \in P_l' \wedge r \in I_1(t) \cap I_1(t') : \tau_{S_0}(t') \geq \tau_{F_0}(t) + \tau_{r_0}(r, Se_l(t,r), Sb_l(t',r)),$$

where:

$P_l' \subseteq P_l$ is the union of all resource task chains.

$\tau_{t_0} : \mathcal{T}_1 \rightarrow \mathbb{R}^+$ gives the duration of a certain task, taking into account the behavioral restrictions imposed by the task as well as the resources involved with the task.

$\tau_{r_0} : \mathcal{R} \times \mathcal{S} \times \mathcal{S} \rightarrow \mathbb{R}^+$ gives the duration of a resource setup from some state to another state, taking into account the behavioral restrictions imposed by the resource. For further information on τ_{t_0} and τ_{r_0}, see "Design of Supervisory Machine Control" referred to above.

In complex manufacturing machines, alternatives exist with respect to different elements of $\mathcal{D}_1$. First of all, alternatives exist concerning precedences. Whereas the precedence relation in level 1 ensures chains of tasks per resource, this order is not essential.. What is essential is that the tasks involved in the manufacturing process of the different manufacturing entities are performed in the right order per manufacturing instance. Furthermore, there is a mutual exclusive constraint: a resource can perform only one task at a time. Given the essential order imposed by the manufacturing process and the mutual exclusive constraint, several possibilities exist to interweave the different manufacturing entities. The alternatives with respect to precedences are analogous to the job shop scheduling (JSS) problem described in M. Pinedo, "Scheduling: Theory, Algorithms, and Systems", Prentice Hall, Englewood Cliffs, (1995).

Second, alternatives exist concerning involved resources. In some cases, multiple resources of the same kind are present in a machine. Several resources can be chosen from to involve in or allocate to a certain task. This is also the case in the generalized job shop scheduling problem described in M. Wennink, "Algorithmic Support for Automated Planning Boards", PhD Thesis, Eindhoven University of Technology, (1995).

Finally, alternatives exist concerning tasks. This is implied by the fact that in some cases multiple tasks exist in a system that have an equivalent effect on the manufacturing process. For instance, multiple paths to transport material from one place to another, or a set of (*m*) tasks of which only a subset (*n*) has to be selected. To determine the required expressiveness of the elements in $\mathcal{D}_2$ to outline the room for choices concerning tasks, some examples from a substrate scanner are analyzed. Regarding the exposure scanning of dies, a 1 out of 2 expressivity is required: a die can be exposed (scanned) in two directions, of which one must be selected. This is analogous to the Rural Postman Problem: *n* out of *m*, in which *n* =1, and $m = 2$ (see E.L. Lawler, J.K. Lenstra, A.H.G. Rinnooy Kan, D.B, Shmoys. "The Traveling Salesman Problem: A Guided Tour of Combinatorial Optimization", Chichester, Wiley-Interscience, pp. 177-178, (1985)). Regarding the measuring of marker pairs on a substrate, the requirements are more difficult. From the (*m*) marker pairs on a substrate, a minimum number (*n*) must be measured. A marker pair consists of several markers, each requiring a measure scan task in either direction (n_0=1). In this case the task selection can be defined as a selection out of *m*, in which the number n_0 of selected marker pairs must be an allowed number, and in which for all markers in the selected marker pairs one scan task is selected. From this, it can be concluded that nesting is needed (n_0 in *n*) and that the allowed number of alternatives can be more than one number $(n \in \mathcal{P}(\mathbb{N}^+))$. Furthermore, a complex machine may contain buffer places. At certain points in the manufacturing process, it is possible to buffer a manufacturing entity. To define this possibility in an intuitive way, it must be possible to describe the fact that also no buffering is allowed, or: 0 can also be an allowed number $(n \in \mathcal{P}(\mathbb{N}))$.

To outline the room for alternatives to select from, literature is followed as much as possible. To outline essential precedences, it is not necessary to introduce a new definition element. Similar to JSS, the precedence element is stripped with respect to level 1 to contain only the essential precedence relations: $P_2 \subseteq P_1$. The stripped precedence instances $P_l \setminus P_2$ concern the scheduled interweaving of the instances that are the result of selection B. The precedence alternatives are outlined by P_2 together with a constraint assuring mutual exclusiveness. To outline resource involvement, an additional definition element called capability is introduced: $\mathcal{C}$, similar to generalized JSS. The involvement function is changed to indicate which capabilities are involved with a certain task: $I_2 : \mathcal{T}_2 \rightarrow \mathcal{P}(\mathcal{C})$. An additional availability function $A : \mathcal{C} \rightarrow \mathcal{P}(\mathcal{R})$ is introduced to describe which resources are available for a certain capability. The resource involvement selection implies selection of one available resource for each involved capability.

To outline the required room for alternatives with respect to tasks, literature can not be followed. Therefore, reasoning is started at the basics. In general, tasks and their precedence relation can be visualized by a graph of type 'activity on node'. For the purpose of modeling selection alternatives concerning tasks, a more general node element $\mathcal{N}_2$ is introduced. Equivalent alternatives can consist of a cluster of nodes (nesting). To identify such clusters, an additional node type cluster ($\mathcal{L}_2$) is introduced. Function $Ln_2 : \mathcal{L}_2 \rightarrow \mathcal{P}(\mathcal{N}_2)$ is introduced to define which nodes belong to which cluster. Furthermore, it is possible that multiple numbers of alternative nodes are allowed to be selected, including the possibility to select none of them. To be able to define which nodes belong to such a group of alternatives, a node type group is introduced: $\mathcal{G}_2$. Function Gn_2 is introduced to define which nodes are in which group, whereas function Ga_2 is introduced to define how many of these nodes are allowed to be selected. The newly introduced definition elements, together with some selection constraints outline the room for selections with respect to tasks. The resulting model can express $[n_1 \text{out of } m]$ or $[n_2 \text{out of } m]$, etc., (aggregates of) tasks, which is abbreviated to $[\{n_1, n_2, \ldots n_x\} \text{ out of } m]$.

Summarizing this analysis, an instantiated, unselected TRS $\mathcal{D}_2$ can be defined by a 14—tuple $(\mathcal{T}_2, \mathcal{L}_2, \mathcal{G}_2, \mathcal{N}_2, \mathcal{R}, C, I_2 A, P_2, Ln_2, Gn_2, Ga_2, Sb_2, Se_2)$:

$\mathcal{T}_2$ is a finite set whose elements are called tasks.

$\mathcal{L}_2$ is a finite set whose elements are called clusters.

$\mathcal{G}_2$ is a finite set whose elements are called groups.

$\mathcal{N}_2$ is a finite set whose elements are called nodes and is a generalization of the model elements mentioned earlier: $\mathcal{N}_2 = \mathcal{T}_2 \cup \mathcal{L}_2 \cup \mathcal{G}_2$.

$\mathcal{R}$ is a finite set whose elements are called resources.

C is a finite set whose elements are called capabilities.

$I_1 : \mathcal{T}_2 \rightarrow \mathcal{P}(C)$ gives the set of capabilities that are involved with a certain task.

$A : C \rightarrow \mathcal{P}(\mathcal{R})$ gives the set of resources that are available for a certain capability.

$P_2 \in \mathcal{P}(\mathcal{N}_2 \times \mathcal{N}_2)$ is the precedence relation between nodes.

$Ln_2 : \mathcal{L}_2 \rightarrow \mathcal{P}(\mathcal{N}_2)$ gives the set of nodes that are in a certain cluster.

$Gn_2 : \mathcal{G}_2 \rightarrow \mathcal{P}(\mathcal{N}_2)$ gives the set of nodes (alternatives) that a group consists of.

$Ga_2 : \mathcal{G}_2 \rightarrow \mathcal{P}(\mathbb{N}))$ gives the allowed numbers (including 0) of nodes to be selected from a group.

$Sb_2, Se_2 : \mathcal{T}_2 \times C \rightarrow S$ gives the begin and the end (physical) states of each capability involved in a certain task.

These definition elements are depicted as a UML class diagram in Figure 4b.

Constraints that have to be satisfied for the instances of the model elements are as follows:

2a: P_2 contains no cycles (see 1b).

2b: For every capability involved in a task, the begin and the end states must be defined (see 1c).

2c: There is no group that has only 0 as allowed number of selected nodes:

$$\nexists g : g \in \mathcal{G}_2 : Gn_2(g) = \{0\} .$$

2d: Nodes which are element of a group have no preceding or succeeding nodes:

$$\forall g, n : g \in \mathcal{G}_2, n \in Gn_2(g) : (\nexists n' : n' \in \mathcal{N}_2 : (n', n) \in P_2 \vee (n, n') \in P_2 .$$

2e: Precedences do not cross group boundaries.

By making choice B, a TRS definition of class 2 is transformed to a TRS definition of class 1. One of the choices has to do with selection from alternatives with respect to tasks. We define a node to be selected if at least one of the tasks that is in it is selected. Let $\mathcal{N}_1$ be the set of selected nodes, then the constraints for $f_B(\mathcal{D}_2)$ can be formulated as follows:

b1: The sequence of selected tasks per resource is a chain (equals 1a).

b2: For each selected task, an available resource must be selected for each involved capability:

$$\forall t, r, c : t \in \mathcal{T}_1, r \in I_1(t), c \in I_2(t) : r \in A(c) .$$

The begin and end state definition is copied from the capability to the selected resource.

b3: Precedence relations are inherited.

$$\forall n, n', t, t' : n, n' \in \mathcal{N}_1, t, t' \in \mathcal{T}_1, t \in TinN(n), t' \in TinN(n'), (n', n) \in P_2 : (t', t) \in P_1 ,$$

where function $TinN(n)$ returns all tasks in a certain node n:

$$TinN(n) = \begin{cases} \{n\} & \text{if } n \in \mathcal{T}_2 \\ (\cup n' : n' \in Ln_2(n) : TinN(n')) & \text{if } n \in \mathcal{L}_2 \\ (\cup n' : n' \in Gn_2(n) : TinN(n')) & \text{if } n \in \mathcal{G}_2 \end{cases}$$

b4: Any node that is not part of another node must be selected, except for groups for which choosing nothing is allowed ('nilgroups'):

$$\forall n,\ell,g : n \in \mathcal{N}_2, \ell \in \mathcal{L}_2, g \in \mathcal{G}_2, n \notin Ln_2(\ell), n \notin Gn_2(g), n \notin \{g' \in \mathcal{G}_2 | 0 \in Ga_2(g')\} : n \in \mathcal{N}_1$$

b5: Any node that is part of a selected cluster must be selected, except for nilgroups:

$$\forall \ell : \ell \in \mathcal{L}_2 \cap \mathcal{N}_1 : \{n \in Ln_2(\ell) | n \notin \{g' \in \mathcal{G}_2 | 0 \in Ga_2(g')\}\} \subseteq \mathcal{N}_1$$

b6: In case nilgroups are not considered, the constraint for groups would be that the number of selected nodes that are part of a selected group must be allowed: $\forall g : g \in \mathcal{G}_2 \cap \mathcal{N}_1 : \#(Gn_2(g) \cap \mathcal{N}_1) \in Ga_2(g)$. The presence of unselected nilgroups in a group relaxes this constraint somewhat, as unselected nilgroups may either or not be counted regarding the allowed number of selected nodes in a group. Knowing this, the constraint can be described as follows:

$$\exists a \in Ga_2(g) : \#(Gn_2(g) \cap N_1) \le a \le \#(Gn_2(g) \cap \mathcal{N}_1) + \#\{g' \in \mathcal{G}_2 \cap Gn_2(g) \mid 0 \in Ga_2(g') \wedge g' \notin \mathcal{N}_1\}$$

To complete the formal definition of the optimization problem, a goal function $f_g : \mathcal{D}_\mathcal{T} \rightarrow R$ is defined, that quantifies the quality of a certain temporal behavior. Examples of factors that play a role in this function are total system duration and number of tasks. Furthermore, two sets of valid functions for f_A and f_B are introduced, F_A, and F_B, respectively. With this, the optimization problem can be described as follows:

$$\max f_A, f_B : f_A \in F_A, f_B \in F_B : f_g(f_A(f_B)\mathcal{D}_2)) \qquad (1.1)$$

In a practical embodiment of the invention, the run-time usability requirement has important consequences. To avoid the machine being idle while waiting for its controller to compute 'optimal' schedules, the algorithm according to an embodiment of the invention is such that tasks can be dispatched to start execution with very small time delays. To achieve this, the schedule is determined in a constructive way, which means from the begin to the end in schedule time. This approach is also safe with respect to extendibility towards handling a TRS definition of level 3. Furthermore, it is possible to dispatch a partial schedule after each task that is added to it. To ensure that the dispatched schedule is an acceptable one, heuristic filters are used to direct the scheduling choices involving choice B. Note that if the algorithm is interrupted to dispatch the schedule, sub-optimal schedule solutions are taken for granted in favor of just getting the machine to work, and non-repetitive behavior might result. Moreover, heuristic filters can be configured such that behavior of a state-based control architecture is copied, which is convenient for software migration purposes. Concerning choice A, the duration of tasks and setup resource state transitions between tasks is determined using dedicated mathematical functions for efficiency and embedability in SMC. Furthermore, the default heuristic of the approach with respect to selection A is to schedule a selected task 'As Soon As Possible' (ASAP), resulting in an 'active' schedule. For memory efficiency, a compact data structure is applied to store the result of selection B that is also compatible with the constructive and ASAP scheduling heuristic of selection A: a heap of pieces (see G.X. Viennot. "Heaps of Pieces, I: Basic Definitions and combinatorial lemmas," Combinatoire Enumerative, Labelle and Leroux, Eds., no. 1234 in Lect. Notes in Math., New York: Springer, pp. 321-350,(1986)). A piece defines a selected task and the selected involved resources, whereas the sequence of pieces in the heap defines the selected precedence relation. Depending on the goal function, a post-processing step is done with respect to selection A to postpone some tasks in order to improve the schedule. Subsequently, other choices with respect to selection B are considered. Taking the run-time aspect into account, the approach explores other alternatives at the beginning of the schedule first, as these tasks will be dispatched first. In Figure 5, the approach is depicted in a flow chart. Summarizing, the approach is a constraint-guided heuristic search algorithm (see M. Pinedo. "Scheduling: Theory, Algorithms, and Systems" referenced above) with a lot of 'escape' points to dispatch work early if desired.

The first step of the method is to determine eligible tasks (pieces). During transformation functionality B, selected alternatives are stored in a heap of pieces. Piece p describes which task $t \in \mathcal{T}_2$ is selected, and the selected resources $rr \subseteq \mathcal{R}_2$ involved: $p \in \mathcal{T}_2 \times \mathcal{P}(\mathcal{R}_2)$. The sequence of pieces in the heap $h \in \mathcal{P}((\mathcal{T}_2 \times \mathcal{P}(\mathcal{R}_2))^*)$ describes the selected precedence relations in addition to the precedence relations in P_2 (intrinsically satisfying constraint b1). If no alternatives with respect to tasks are taken into account, considering a heap *h* containing passed (= selected up to then) tasks $t_p \subseteq \mathcal{T}_2$, a next piece *p* consisting of task *t* and involved resources *rr* is eligible to form an extended heap *hp* if and only if:

- the predecessors of t are in pieces of heap h, and t is not:

$$Et(h) = \{t \in \mathcal{T}_2 \setminus t_p | (\forall t' : (t', t) \in P_2 : t' \in t_p\} \tag{1.2}$$

- constraint b2 is satisfied concerning rr:

$$\mathrm{Er}(t) = \{rr \subseteq \mathcal{R} | (\forall r, c : r \in rr, c \in I_2(t) : r \in A(c))\} \tag{1.3}$$

- The set of eligible next pieces for sequence h can be defined as follows:

$$E(h) = \{(t, rr) | t \in Et(h) \wedge rr \in Er(t)\} \tag{1.4}$$

The set of feasible heaps $\mathcal{H} \subseteq \mathcal{P}((\mathcal{T}_2 \times \mathcal{P}(\mathcal{R}_2))^*)$ can be defined by induction as follows:

$$\varepsilon \in \mathcal{H} \tag{1.5}$$

$$\varepsilon \in \mathcal{H} \wedge p \in E(h) \Rightarrow hp \in \mathcal{H} \tag{1.6}$$

In (1.5), ϵ denotes the empty heap.

Considering the alternatives with respect to tasks, function *Et(h)* must be extended. During the selection process, tasks that are not selected and will not be selected anymore are called 'bypassed'. After the selection process, the set of bypassed tasks equals $\mathcal{T}_2 \setminus \mathcal{T}_1$. Function *Et(h)* needs to consider only tasks that are neither passed nor bypassed. For the tasks that are neither passed nor bypassed, the predecessor relation must be checked. In case no alternatives with respect to tasks are considered, all predecessors must be in the heap (see Equation 1.2). This condition is relaxed in case of predecessors of type group; all (possibly inherited, see constraint b3) predecessors must be 'succeedable'.

Let function $anc : \mathcal{N}_2 \to \mathcal{P}(\mathcal{N}_2)$ be a recursive function that determines the ancestors of a node, which are those nodes in which a node n is contained:

$$anc(n) = (\cup n' : n' \in \mathcal{N}_2 \setminus \mathcal{T}_2 \wedge n \in Gn_2(n') \cup Ln_2(n') : \{n'\} \cup anc(n')) \tag{1.7}$$

This recursion is finite as the nodes have a hierarchical structure, which is explored upwards only in this function.

Let function $succ : \mathcal{N}_2 \to \mathcal{P}(\mathcal{N}_2)$ be a function that determines the successors of a node n:

$$succ(n) : (\cup n' : n' \in \mathcal{N}_2 \wedge (\exists n'' : n'' \in anc(n) \cup \{n\} : (n'', n') \in P_2) : \{n'\}) \tag{1.8}$$

A task is succeedable when it is passed, and a cluster is succeedable when all nodes in it are succeedable. The non-succeedable nodes of a succeedable group may contain no passed tasks. Furthermore, if none of the nodes of a group is succeedable whereas zero is an allowed number, a group is succeedable when all of its predecessors are succeedable. In other cases, a group is succeedable when the (non zero) number of succeedable nodes of it is an allowed number.

Let $ns : \mathcal{N}_2 \to \mathbb{B}$ be a recursive function that determines whether a node n is succeedable:

$$\begin{aligned}
ns(n) = \quad & (n \in \mathcal{T}_2 \wedge n \in t_p) \\
\vee \quad & (n \in \mathcal{L}_2 \wedge (\forall n' : n' \in Ln_2(n) : ns(n'))) \\
\vee \quad & (n \in \mathcal{G}_2 \wedge (\forall n' : n' \in Gn_2(n) \wedge \neg ns(n') : (\forall t : t \in \mathcal{T}_2 \wedge n \in anc(t) : t \notin t_p)) \wedge \\
& ((\nexists n' : n' \in Gn_2(n) \wedge ns(n')) \wedge (0 \in Ga_2(n))) \wedge (\forall n' : n' \in \mathcal{N}_2 \wedge (n', n) \in P_2 : ns(n')) \\
& \vee (((\exists n' : n' \in Gn_2(n) \wedge ns(n')) \vee (0 \notin Ga_2(n))) \wedge (\left|\{n' | n' \in Gn_2(n) \wedge ns(n')\}\right| \in Ga_2(n))) \\
&)
\end{aligned} \tag{1.9}$$

This recursion is finite as the nodes have a hierarchical structure which is explored downwards only, and precedences have no loops and are explored backwards only. The set of succeedable nodes, n_s, can be defined as follows: $n_s = \{n \in \mathcal{N}_2 | ns(n)\}$.

Let n_i be the set of initiated nodes. A node is initiated if it is not succeedable and contains a passed task. This set can be defined as follows:

$$n_i = (\cup n : n \in \mathcal{N}_2 \setminus n_s \wedge (\exists t : t \in t_p : n \in anc(t)) : \{n\}) \tag{1.10}$$

A task is bypassed when it is not passed and when it is in a (node of a) group that is not succeedable or initiated whereas the maximum number of nodes of the group is succeedable or initiated, or if any succeeding node of it is succeedable. The set of bypassed tasks, t_b, can be defined as follows.

$$\begin{aligned}
t_b = \quad & \{ \; t \in \mathcal{T}_2 \setminus t_p \\
& \;|\; (\exists g : g \in \mathcal{G}_2 : Gn_2(g) \cap (anc(t) \cup \{t\}) \setminus (n_s \cup n_i) \wedge |Gn_2(g) \cap (n_s \cup n_i)| = \max(Ga_2(g))) \\
& \vee (\exists t' \in t_p : (\{t'\} \cup anc(t')) \cap succnil(t) \neq \emptyset) \\
& \}
\end{aligned} \tag{1.11}$$

where function $succnil : \mathcal{T}_2 \to \mathcal{P}(\mathcal{N}_2)$ determines the successors of a task t, including the successors of succeeding nilgroups:

$$succnil(t) = succ(t) \cup (\cup n : n \in (succ(t) \cap \mathcal{G}_2) \wedge 0 \in Gn_2(n) : succnil(n))$$

Using this, function $\mathcal{E}t(h)$ when considering alternatives with respect to tasks can be defined as follows:

$$\mathcal{E}t(h) = \{t \in \mathcal{T}_2 \setminus t_p \setminus t_6 | (\forall n, n' : n \in anc(t) \wedge (n', n) \in P_2 : n' \in n_s)\} \qquad (1.12)$$

The second step of the method is application dependent but the third step can be described generally.

An 'As Soon As Possible' (ASAP) heuristic for the choice concerning timing can be associated with an intuitive interpretation, namely that of a heap of pieces (see G.X. Viennot. "Heaps of Pieces, I: Basic Definitions and Combinatorial Lemmas," Combinatoire Enumerative, Labelle and Leroux, Eds., no. 1234 in Lect. Notes in Math., New York: Springer, pp. 321-350, (1986)). Timing behavior of a TRS can be visualized using a Gantt chart. When a Gantt chart is turned 90° counter-clockwise, the resource occupation by tasks can be interpreted as a heap of pieces $p \in \mathcal{T}_2 \times \mathcal{P}(\mathcal{R})$. The first element of this tuple p_0 equals the considered task, whereas the second element p_1 equals the resources involved with this task: $p = (t, I_1(t))$. Resources can be associated with the slots on the horizontal axis, whereas (task duration) time is represented on the vertical axis. Tasks are represented by rectangular pieces. The task duration τ_{t_0} is represented by the height of a rectangle, whereas the involved resources are represented by its 'width'. The 'ASAP' heuristic can be associated with pieces falling onto each other under the influence of 'gravity'.

The upper contour of a heap is associated with the time until which the resources are occupied by the pieces in the heap. It is defined as the R-dimensional row vector $u_{\mathrm{H}}(h)$, where $u_{\mathrm{H}}(h,r)$ is the height of the heap on slot *r*. The upper contour state is defined as the R-dimensional row vector $u_{\mathrm{Hs}}(h)$, where $u_{\mathrm{Hs}}(h,r)$ is the (physical) state of resource *r* at time $u_{\mathrm{H}}(h,r)$.

The horizontal ground convention (see constraint a2), which can be associated with time starting at 0, yields:

$$u_H(\varepsilon) = (0,...,0) \qquad (1.13)$$

The upper contour of heap *hp* that results after piling up a piece *p* on top of a heap *h* is equal to the finish time of task *t* for the resources that are occupied by *t* and equal to the upper contour of *h* for the other resources:

$$u_{\mathrm{H}}(hp,r)=\begin{cases}\tau_{\mathrm{F}_0}(p.0,h) & \text{if } r\in p.1\\ u_{\mathrm{H}}(h,r) & \text{if } r\notin p.1\end{cases}\tag{1.14}$$

The upper contour state of heap *hp* that results after piling up a piece *p* on top of a heap *h* is equal to the end state of task *t* for the resources that are occupied by *t* and equal to the upper contour state of *h* for the other resources:

$$u_{\mathrm{Hs}}(hp,r)=\begin{cases}\mathrm{Se}_1(p.0,h) & \text{if } r\in p.1\\ u_{\mathrm{Hs}}(h,r) & \text{if } r\notin p.1\end{cases}\tag{1.15}$$

The finish time of a task *t* is obtained by adding its duration to its start time (see constraint a2):

$$\tau_{\mathrm{F}_0}(t,h)=\tau_{S_0}(t,h)\ +\tau_{t_0}(t)\tag{1.16}$$

The start time of a task *t* associated with piece *p* being piled upon top of a heap *h* is influenced by two components: by its preceding tasks (see constraint a3) on the one hand and by the setup state transitions of the involved resources (see constraint a4) on the other. Note that this precedence constraint is an extension of that described in S. Gaubert and J. Mairesse, "Task Resource Models and (max, +) Automata," in Idempotency, Cambridge, U.K. Cambridge University Press, pp. 133-144, 1998. It can be determined by taking the highest value of either the highest finish time of its preceding tasks, $\tau_{S_{0_p}}(t,h)$, or of the highest part of the upper contour of the heap *h* beneath the piece after setup of the involved resources, $\tau_{S_{0_r}}(t,h)$:

$$\tau_{S_0}(t,h)=\max(\tau_{S_{0_p}}(t,h),\tau_{S_{0_r}}(t,h))\tag{1.17}$$

Regarding preceding tasks, the start time of task *t* equals the maximum finish time of the passed tasks that precede any node *n* that *t* is in:

$$\tau_{S_0}(t,h)=\max t':t'\in(\cup n:n\in\mathcal{N}_2\wedge t\in(\mathrm{TinN}(n):tpp(n,h))):\tau_{F_0}(t',h)\tag{1.18}$$

where function $tpp:\mathcal{N}_2\times(\mathcal{T}_2\times\mathcal{P}(\mathcal{R}))^*\to\mathcal{P}(\mathcal{T}_2)$ determines all passed tasks preceding a node. If any of these predecessors is an unselected nilgroup, the passed tasks in the predecessors of that nilgroup must be retrieved. Because of this, the function is recursive and is defined as follows:

$tpp(n,h)=$

$$\begin{cases} \{\} & \text{if } \exists n' : (n',n) \in \mathrm{P}_2 \\ \cup\, n' : (n',n) \in \mathrm{P}_2 : \begin{cases} \mathrm{t_p} \cap \mathrm{TinN}(n') & \text{if } \exists t' : t' \in t_p \cap \mathrm{TinN}(n') \\ tpp(n',h) & \text{if } \nexists t' : t' \in t_p \cap \mathrm{TinN}(n') \end{cases} & \text{if } \exists n' : (n',n) \in \mathrm{P}_2 \end{cases} \tag{1.19}$$

Regarding resource setup, the start time of a task t can be obtained as follows:

$$\tau_{\mathrm{S_0}r}(t,h) = \max_{\mathrm{r} \in I_1(t)} (u_{\mathrm{H}}(h,r) + \tau_{r_0}(r, u_{\mathrm{Hs}}(h,r), Sb_1(t,r))) \tag{1.20}$$

The above general situation can be applied to a dual stage lithographic apparatus as described above. By way of an example, 3 lots each of 5 substrates of a double exposure recipe are considered. For each lot, different masks are needed: masks 1 and 2 for the first lot, masks 3 and 4 for the second lot and masks 5 and 6 for the third lot. Each substrate consists of 171 dies of size 26x13mm^2, in which the 13mm is in the scanning direction. Prior to the exposure of the dies, each substrate must be measured. This measuring step involves measurement of alignment marker pairs. 25 alignment marker pairs are placed on each substrate, of which at least 16 pairs must be measured in either direction of each marker to reach the required minimum manufacturing accuracy.

For the purpose of this example, only the selection of tasks and their order is concerned.

Alternatives lie in the following areas:

1. In which order and direction to execute the exposures of the dies of each substrate,
2. Which marker measuring tasks to execute in which order,
3. Which mask handling tasks to execute in which order.

In Figure 6, which depicts the mask handling arrangements of the lithographic apparatus of Figure 1 in more detail, the sequential mask handling tasks that can be executed for one mask are indicated using an arrow, and are labeled by their sequence number. The dotted arrows concern usage of a buffer (tasks MTa6 and MTa7), which is optional. Each resource is denoted by a square.

For illustration, the timing behavior resulting from two different settings for the heuristic filters is described.

The description of the first setting, setting I, is as follows:

1. The exposure sequence is a horizontal meander.
2. The minimum number (16) of marker pairs is measured.

3. Preprocessing of masks is started as soon as possible. If masks can not go through to the mask stage, they are put in the buffer (IRL).

In Figure 7, the ASAP time behavior for setting I is depicted in a Gantt chart. In this chart, tasks are colored per substrate or mask, exposure tasks get the color of the mask. In Figure 8, the critical path of the time behavior is depicted in a Gantt chart.

The description of the second heuristic setting, setting II, is as follows:

1: The exposure sequence is a vertical meander.

2: All marker pairs (25) are measured.

3: Preprocessing of masks is started only if less than 4 preprocessed masks are available for coming exposures. If the robot can choose to either put a mask on the turret or start preprocessing a next mask, it chooses to put the mask on the turret. Masks are put in the IRL buffer after preprocessing only if they have to wait and the coming lot requires another mask that is not preprocessed yet.

In Figure 9, the resulting time behavior using setting II is depicted in a Gantt chart. The time needed for manufacturing the three lots is decreased by more than 5% compared to setting I. Half of this is caused by the changed exposure sequence that itself decreased its duration by about 10%. The other half is caused by changed mask handling. As a side effect, better product quality is achieved as more marker pairs are measured. This does not cost any time, as measuring still is not on the critical path, as can be seen in Figure 10.

An embodiment of the present invention therefore provides a formal basis for model-based supervisory control of manufacturing machines starting from an instantiated, unselected TRS definition. To define the TRS behavior optimization problem formally, the choice making functionalities A and B are regarded as transformation functions. The same goes for the quantification of the quality of the behavior resulting from the choices made, which is done using a goal function. The constraints that have to be satisfied for transformations A and B are defined mathematically. With this, the behavior optimization problem is defined as the maximum value of the goal function for which the constraints concerning transformations A and B are satisfied.

An instantiated, unselected TRS in the domain of complex manufacturing machines leaves room for choices (selection B) with respect to tasks, resources and task order. To define the alternatives to choose from, the generalized JSS approach (see "Algorithmic Support for

Automated Planning Boards" referenced above) is followed concerning alternatives with respect to resources and task order, whereas a new approach is developed concerning alternatives with respect to tasks. This approach is more expressive than previous approaches.

Choice functionality A and B are combined in an optimization approach that is suited for run-time application in supervisory machine control. The basis of the approach is a top-down iterative heuristically filtered beam search algorithm and that is coupled with the available search time to combine good behavior quality with little control overhead. The algorithm is extendable for machine-specific issues, which can be defined as additional selection constraints.

A second embodiment of the invention is applied to a lithographic apparatus employing extreme ultraviolet (EUV) radiation in the projection beam. As this radiation is absorbed by air, exposure must take place in vacuum, whereas the apparatus resides in atmospheric pressure. To bring the substrates from atmospheric pressure to vacuum two load locks L0, L1 are provided. To transport the substrates between the different subsystems two robots R0, R1 are provided. A schematic layout of the apparatus is depicted in Figure 11.

In Figure 11, the parallel mechatronic systems – track T0, load locks L0, L1, robots R0, R1, aligners A0, A1 and substrate tables S0, S1 (also referred to below as stages) – considered are depicted by a circle, and the possible transport paths are depicted by arrows. Each mechatronic system (except the track) can carry only one substrate, which is depicted between brackets. The tight layout of the apparatus makes it possible for the robots to collide if they both move from or to a lock, which is depicted by the cross-hatched area. The apparatus is a dual-stage apparatus with separate measure and expose stations (not shown in Figure 11). In the measure station, substrates can be loaded onto and unloaded from a substrate table at their load and unload position, respectively. Each resource can reach a limited number of states.

The state-transition diagrams or automata of the different mechatronic systems are shown in Figures 12 to 16. In these Figures, the initial state is denoted by an extra circle, and transitions are labeled with a time duration or a task name, which will be explained later. Switching of areas of the tables must be done synchronously to avoid collision of the substrate stages: chuck swap. This is depicted by the dashed connections between the diagrams of the two tables in Figure 12. As an example, a work to be scheduled concerns a typical batch (lot) of 15 substrates.

As discussed earlier, a scheduling problem can be associated with an instantiated, unselected TRS definition (level 2) in Figure 4. A job shop scheduling problem can be defined by a 6—tuple ($\mathcal{T}_2$, $\mathcal{R}$, I_2, P_2, Sb_2, Se_2) in which:

$\mathcal{T}_2$ is a finite set whose elements are called tasks.

$\mathcal{R}$ is a finite set whose elements are called resources.

I_2: $\mathcal{T}_2 \rightarrow \mathcal{R}$ gives the resource that is involved in a certain task.

$P_2 \subseteq \mathcal{T}_2 \times \mathcal{T}_2$ is the precedence relation between tasks.

$Sb_2, Se_2 : \mathcal{T}_2 \times \mathcal{R} \rightarrow \mathcal{S}$ give the begin and the end (physical) state of the resource involved in a certain task.

Note that, by convention, the definition level is added to each definition element in subscript. Definition elements which are not level specific have no suffix.

Constraints that have to be satisfied for the instances of the definition elements are as follows:

2a P_2 contains no cycles.

2b For every resource involved in a task, the begin and the end state must be defined.

However, in a complex machine multiple resources can exist which all are capable of the same work. Furthermore, some tasks involve synchronous transitions of multiple resources. As this can also be the case in job shops, the job shop scheduling problem has been generalized to incorporate this issue (see "Algorithmic support for automated planning boards " referenced above)

A generalized job shop scheduling problem can be defined by a 8—tuple $(\mathcal{T}_2, \mathcal{R}, C, I_2, A, P_2, Sb_2, Se_2)$:

$\mathcal{T}_2$ is a finite set whose elements are called tasks.

$\mathcal{R}$ is a finite set whose elements are called resources.

C is a finite set whose elements are called capabilities.

$I_2 : \mathcal{T}_2 \rightarrow \mathcal{P}(C)$ gives the set of capabilities that are involved in a certain task.

$A : C \rightarrow \mathcal{P}(\mathcal{R})$ gives the set of resources that are available for a certain capability.

$\mathcal{P}_2 \subseteq \mathcal{T}_2 \times \mathcal{T}_2$ is the precedence relation between tasks.

$Sb_2, Se_2 : \mathcal{T}_2 \times C \rightarrow S$ give the begin and the end (physical) state of each capability involved in a certain task.

Constraints that have to be satisfied for the instances of the definition elements are as follows:

2a P$_2$ contains no cycles.

2b For every capability involved in a task, the begin and the end state must be defined.

By selecting, which comes down to the assignment of resources to tasks, and determination of the order in which tasks are executed per resource, an unselected TRS is transformed into a selected, untimed TRS, which can be defined by a 6-tuple $(\mathcal{T}_1, \mathcal{R}, I_1, P_1, Sb_1, Se_1)$:

$\mathcal{T}_1$ is a finite set whose elements are called tasks.

$\mathcal{R}$ is a finite set whose elements are called resources.

$I_1 : \mathcal{T}_1 \rightarrow \mathcal{P}(\mathcal{R})$ gives the set of resources that are involved in a certain task.

$P_1 \subseteq \mathcal{T}_1 \times \mathcal{T}_1$ is the precedence relation between tasks.

$Sb_1, Se_1 : \mathcal{T}_1 \times \mathcal{R} \rightarrow S$ give the begin and the end (physical) states of each resource involved in a certain task.

Constraints that have to be satisfied for the instances of the definition elements are as follows:

1a The sequence of tasks per resource is a chain.

Constraints that have to be satisfied for the selecting transformation can be formulated as follows:

b1 The sequence of selected tasks per resource is a chain (equals 1a).

b2 For each selected task, an available resource must be selected for each involved capability:

$$(\forall t, r, c : t \in \mathcal{T}_1, r \in I_1(t), c \in I_2(t) : r \in A(c)).$$

By timing, a selected, untimed TRS is transformed into a timed TRS, which can be defined by a 5-tuple $(\mathcal{T}_0, \mathcal{R}, I_0, \tau_{S_0}, \tau_{F_0})$:

$\mathcal{T}_0$ is a finite set whose elements are called tasks.

$\mathcal{R}$ is a finite set whose elements are called resources.

$I_0 : \mathcal{T}_0 \rightarrow \mathcal{P}(\mathcal{R})$ is the set of resources that are involved in a certain task.

$\tau_{S_0}, \tau_{F_0} : \mathcal{T}_0 \rightarrow \mathbb{R}^+$ are the start time and the end time of a certain task, which implies that these are the same for all involved resources.

Furthermore, note that a timed TRS can be visualized as a Gantt chart.

The constraint that has to be satisfied for the instances of the definition elements is as follows:

0a Per resource there is a chronological sequence of pairs of task start and task finish times.

Constraints that have to be satisfied for the timing transformation are as follows:

a1 Nothing changes with respect to tasks and the involved resources:

$$\mathcal{T}_0 = \mathcal{T}_1, I_0 = I_1$$

a2 By convention, time starts at 0. Furthermore, the finish time of a task equals its start time plus its duration:

$$(\forall t : t \in \mathcal{T}_1 : \tau_{S_0}(t) \geq 0 \wedge \tau_{F_0}(t) = \tau_{S_0}(t) + \tau_{t_0}(t)).$$

a3 For subsequent tasks, the start time of the succeeding task is greater than or equal to the finish time of the preceding task:

$$(\forall t, t' : (t, t') \in P_1 : \tau_{S_0}(t') \geq \tau_{F_0}(t)).$$

a4 To match the states of subsequent tasks on the same resource, setup state transitions of the resource might be necessary. In these cases, the start time of the succeeding task is greater than or equal to the finish time of the preceding task plus the duration of the setup resource state transition between the tasks:

$$(\forall t', r : (t, t') \in P_1' \wedge r \in I_1(t) \cap I_1(t') : \tau_{S_0}(t') \geq \tau_{F_0}(t) + \tau_{r_0}(r, Se_1(t, r), Sb_1(t', r))),$$

where:

$P_1' \subseteq P_1$ is the union of all resource task chains.

$\tau_{t_0} : \mathcal{T}_1 \rightarrow \mathbb{R}^+$ gives the duration of a certain task, taking into account the behavioral restrictions imposed by the task as well as the resources involved with the task.

$\tau_{r_0} : \mathcal{R} \times \mathcal{S} \times \mathcal{S} \rightarrow \mathbb{R}^+$ gives the duration of a resource setup from some state to another state, taking into account the behavioral restrictions imposed by the resource. For further information on τ_{t_0} and τ_{r_0} (see "Design of Supervisory Machine Control" referenced above).

The mapping of the scheduling problem of this embodiment onto the definition of a generalized job shop scheduling problem can be split into two sections: system-dependent elements and work-dependent elements. The system-dependent elements can be defined as follows:

There are 5 capabilities: stage, robot, aligner, lock and track:

$$\mathcal{C} = \{S, R, A, L, T\}.$$

There are 9 resources: stage0, stage1, robot0, robot1, aligner0, aligner1, lock0, lock1, track0:

$$\mathcal{R} = \{S0, S1, R0, R1, A0, A1, L0, L1, T0\}.$$

The available resources for each capability are defined as follows:

$$\mathcal{A} = \{(S,\{S0, S1\}), (R, \{R0, R1\}), (A, \{A0, A1\}), (l, \{L0, L1\}), (T, \{T0\})\}$$

To define the work-dependent elements, the sequence of tasks per substrate through the apparatus (life of a substrate) is analyzed. As shown in Figure 17, first, a substrate is transported from the track into a lock (T2L). Subsequently, the pressure is pumped down (PD), and the substrate is transported onto the robot (L2R). The robot rotates from the lock to the aligner (RLA), places the substrate onto the aligner (R2A), and the alignment takes place (AL). After that, the robot takes the substrate from the aligner (A2R), rotates to the stage (RAS), and places the substrate onto the stage (R2S). On the stage, measurement takes place (MEA), and after stage swap (SW) the substrate is exposed (EXP). Then, the stage swaps to the unload position in the measure area (SW) where the robot takes the substrate from the stage (S2R). The robot rotates to the lock (RSL), and puts the substrate in the lock (R2L). Finally, the lock pumps up the pressure (PU) and the substrate is taken from the lock by the track (L2T). The life of a substrate can be defined by $\mathcal{T}_2$ and P_2, and can graphically displayed by a task graph, as is shown in Figure 17. A first attempt for the definition of the task graph for the entire batch could be 15 of these identical sequences.

However, when looking closer it appears that some of the tasks in Figure 17 may be necessary besides the ones in life of a material. For example, if a lock subsequently has to pump down two substrates, it must pump up in between. The same goes for the rotate tasks and the table swap. In job shop scheduling, such tasks are called setups, which implies that they are a consequence of the sequence of regular tasks on the same resource. Depending on the selected sequence of tasks on the same resource, setups are required or not. More particularly: if for some resource the end state of a preceding task does not match the begin state of a succeeding task, a setup is inserted to bridge this gap. This implies that setup tasks can be left out of life of a substrate, as is displayed in Figure 18.

For one substrate, e.g. substrate 1, the work-dependent elements can be defined as follows:

$\mathcal{T}_2$ = {W1-T2L, W1-L2R, ...}

P_2 = {(W1-T2L, W1-L2R), (W1-L2R, W1-R2A), ...}

I_2 = {(W1-T2L, {T L}), (W1-L2R, {L, R}), (W1-R2A, {R, A}), ...}

Sb_2 = {{(W1-T2L, T, rts), (W1-T2L, L, atm)}, {(W1-L2R, L, vac), (W1-L2R, R, @1)}, {(W1-R2A, R, @a), (W1-R2A, A, idle)}, {(W1-AL, A, idle)}, ...}

Se_2 = {{(W1-T2L, T, rtr), ...}

Note that by convention, state names are in lower case whereas task names are in upper case and start with the material associated.

Although the generalized job shop scheduling definition described above forms a good basis for the scheduling problem in a complex machine, some essential constraints are missing to avoid unfeasible schedules. Some of these have to do with material logistics. For instance, whereas material transport is feasible from the track to any of the lock resources of the lock capability, this is not the case from any lock to any robot: logistic flow feasibility. Moreover, if for instance a substrate is transported into one of the locks it must be assured that it is taken from the same lock: logistic flow integrity. Furthermore it must be assured that not too many substrates are in one of the locks at the same time as physical room does not allow for that: material capacity feasibility.

Also resource interference causes additional constraints. Unlike in job shop scheduling, a setup may be constrained to be executed synchronous with other transitions only, for instance the stage swap. On the other hand, multiple setups may be constrained to be executed one at a time as they involve visiting the same hazardous area, for instance robot rotations in front of the locks. Both these complications may be appropriate for part of a setup transition only, for instance the robot rotation from state @a to state @l visits the hazardous area between the intermediate state @ca to state @l only. Therefore, a possibly compound state transition or setup must be decomposed in elementary transitions. The same goes for the state transition of a substrate stage from @e to @lm, which must go via state @u.

Finally, the required nanometer accuracy imposes constraints on some time windows. As the substrate is conditioned on the pre-aligner and the stage but not in between, the time in between should not be more than necessary. This means that tasks A2R and R2S should be executed without delay. Furthermore, the time between exposure and transport to the track (Post Exposure Bake time) should be as constant as possible, to achieve good imaging uniformity.

To ensure feasible machine behavior with respect to material, resources may not be overloaded as they have limited room for material. As material transport tasks also play a role, it must be possible to describe the fact that resources are occupied by material instances and the consequences of tasks for the location of material instances.

If choices with respect to resources did not play a role, and each resource would have room for one material instance, it would be possible to ensure feasible machine behavior without the introduction of the notion of material. A material instance, as well as the physical material location at a resource could be modeled as a resource. However, as material remains residing at a resource after the physical task, additional 'material occupation' tasks are introduced after each physical transport task. These 'material occupation' tasks indicate that a resource is occupied by some material instance, and can only be succeeded by a transport task, after which the material resides on another resource. The 'material occupation' tasks finish when subsequent transport tasks start. This is illustrated in Figure 19.

However, if choices with respect to resources do play a role, the approach presented earlier is deficient. In this case, not just any available resources are allowed to be chosen, as the integrity of the logistic flow of the material must be assured. For example: if two resources S1 and S2 of capability S exist, a material that is transported to resource S1 must also be transported from S1. This implies that resource S2 is not eligible in this case, even though it is available. To describe the logistic integrity issues in an intuitive way, the notion of material is added to TRS definition level 2.

It is assumed that the logistic (material configuration) effect of a task on the involved material instances is the same for all involved material instances.

The following 5 elements are added to the unselected TRS definition $\mathcal{D}_2$:

- $\mathcal{M}$ is a finite set whose elements are called material instances.
- $Cb_2, Ce_2 : \mathcal{T}_2 \rightarrow \mathcal{P}(C \times \mathcal{P}(\mathcal{M})$ give the material configuration at the begin and the end of a task as a set of tuples defining each occupied involved capability and the material instances residing on it.
- Rm: $\mathcal{R} \rightarrow \mathbb{N}$ gives the number of material instances that can reside on a certain resource.
- $Mf \subseteq \mathcal{R} \rightarrow \mathcal{R}$ represents the physically possible material flow as a set of tuples defining from which resource to which resource material can flow.

Additional constraints that have to be satisfied concerning $\mathcal{D}_2$:

2-i The capabilities in Cb_2 and Ce_2 must be consistent with I_2:

$$(\forall t, c : t \in \mathcal{T}_2, c \in C, c, \in \{cm.0 | cm \in Cb_2(t) \cup Ce_2(t)\} : c \in I_2(t))$$

2-ii For each capability involved in a task, the logistic effect of the task is the same for all materials involved (for example, a transport of two sets of materials from the same capability to two different other capabilities is not possible in one task):

$(\forall t : t \in \mathcal{T}_2 : \{cm.1 | cm \in Cb_2(t)\} = \{cm.1 | cm \in Ce_2(t)\})$. This constraint implies that only closed systems are considered, which means that material does not enter or leave the system.

Let $\mathcal{P}_{2m} : \mathcal{D}_2 \times \mathcal{M} \rightarrow \mathcal{P}(\mathcal{T}_2 \times \mathcal{T}_2)$: be a function describing for each material $m \in \mathcal{M}$ in a TRS definition $D_2 \in \mathcal{D}_2$, a precedence relation between related tasks (the material 'life') without redundant edges and with matching capabilities:

$$\begin{aligned} P2m\,(D2, m) = \{ & (t, t') \\ & |(t, t') \in P_2 \\ & \wedge \{cm.0 | cm \in Ce_2(t), m \in cm.1\} = \{cm.0 | cm \in Cb_2(t'), m \in cm.1\} \\ & \wedge \neg redundant(t, t', P_2) \\ & \} \end{aligned} \tag{2.1}$$

Where function $redundant : \mathcal{T}_2 \times \mathcal{T}_2 \times \mathcal{P}(\mathcal{T}_2 \times \mathcal{T}_2) \rightarrow \mathbb{B}$ determines whether a precedence edge (t, t') is redundant in a precedence relation P:

$$redundant(t, t', P) = (\exists t'' : t'' \in \mathcal{T}_2 \wedge t'' \neq t \wedge t'' \neq t' : path(t, t'', P) \wedge path((t'', t' P))$$

(2.2)

And where function $path : \mathcal{T}_2 \times \mathcal{T}_2 \times \mathcal{P}(\mathcal{T}_2 \times \mathcal{T}_2) \to \mathbb{B}$ determines whether there is a path between two tasks t and t' in a precedence relation P:

$$path(t,t',P) = \begin{cases} true & if\ t = t' \\ (\exists t'' : (t,t'') \in P : path(t'',t,\mathrm{P})) & if\ t \neq t' \end{cases} \tag{2.3}$$

The additional constraints that have to be satisfied for transformation B from $\mathcal{D}_2$ into $\mathcal{D}_1$ are as follows:

b-i The resources involved in life of material instance $m \in \mathcal{M}$ in a TRS definition $D_2 \in \mathcal{D}_2$ match (logistic flow integrity):

$$\begin{aligned}(\forall t,t' &: (t,t') \in P_{2m}(D_2,m) \\ &: I_1(t) \cap (\cup m : m \in Ce_2(t).1 : A(Ce_2(t).0)) \\ &: I_1(t') \cap (\cup m : m \in Cb_2(t').1 : A(Cb_2(t').0)) \\ )\end{aligned} \tag{2.4}$$

b-ii The combination of involved resources in material transport is physically possible (logistic flow feasibility):

$$\begin{aligned}(\forall t,m,r_b,r_e &: t \in T_1, m \in \mathcal{M}, r_b, r_e \in \mathcal{R}, \\ &\{r_b\} = \mathrm{I}_1(t) \cap (\cup m : m \in Cb_2(t).1 : A(Cb_2(t).0)), \\ &\{r_e\} = \mathrm{I}_1(t) \cap (\cup m : m \in Ce_2(t).1 : A(Ce_2(t).0)), \\ &: r_b = r_e \vee (r_b, r_e) \in Mf \\ )\end{aligned} \tag{2.5}$$

b-iii The material capacity of a resource is not exceeded (material capacity feasibility). Let $\mathrm{P}_{1r} : \mathcal{D}_1 \times \mathcal{R} \to \mathcal{P}(\mathcal{T}_1 \times \mathcal{T}_1)$ be a function describing for each resource r in a TRS definition $D_l \in \mathcal{D}_1$ a linear precedence relation between related tasks, where linear means that the related tasks form a chain:

$$\begin{aligned}P_{lr}(D_l,r) = \{&(t,t') \\ &|(t,t') \in P_l \\ &\wedge I_l(t) \cap I_1(t') \neq \emptyset \\ &\wedge \neg redundant\ (t,t',P_l) \\ &\}\end{aligned} \tag{2.6}$$

Let tchainr : $\mathcal{P}(\mathcal{T}_1 \times \mathcal{T}_1) \to \mathcal{T}_1^*$ be a function that returns the task chain corresponding with a linear precedence relation P_l.

$$tchainr(P_1) = \begin{cases} \varepsilon & \text{if } P_1 = \emptyset \\ [firstp(P_1).0] + +tchainr(P_1 \setminus \{firstp(P_1)\})] & \text{if } P_1 \neq \emptyset \end{cases} \tag{2.7}$$

Where $a{+}{+}b$ denotes concatenation of sequence a and b into ab.

Where function firstp: $\mathcal{P}(\mathcal{T}_1 \times \mathcal{T}_1) \rightarrow \mathcal{T}_1 \times \mathcal{T}_1$ determines the first precedence edge in a linear precedence relation P_l:

$$firstp(P_1) = \{(t,t')|(t,t') \in P_1 \wedge (\nexists t'': t'' \in \mathcal{T}_1 : (t'',t) \in P_1)\} \tag{2.8}$$

Let $S_m(r,s)$ be the material configuration of resource *r* after execution of task sequence *s*. Before executing any tasks, the material configuration of a resource is given and is defined as the initial material configuration: $S_m(r,\varepsilon) = S_{m\text{-}i}(r)$. The material configuration after execution of task sequence *st* is defined as follows:

$$S_m(r,st) = S_m(r,s) \setminus mat(Cb_2(t),r) \cup mat(Ce_2(t),r) \tag{2.9}$$

Where function $mat : \mathcal{P}(C \times \mathcal{P}(\mathcal{M})) \times \mathcal{R} \rightarrow \mathcal{P}(\mathcal{M})$ returns the material associated with the capability that has the resource available for it.

$$mat(CM,r) = \{m | m \in cm.1 \wedge cm \in CM \wedge r \in A(cm.0)\} \tag{2.10}$$

Then the material capacity constraint for a TRS definition $D_1 \in \mathcal{D}_1$ can be defined as follows:

$$(\forall r, st : r \in \mathcal{R} \wedge t \in \mathcal{T}_1 \wedge st \preceq tchainr(P_{1r}(D_1,r)) : |S_m(r,st)| \leq Rm(r)) \tag{2.11}$$

Additional constraints are introduced that outline valid extension of a selection to avoid invalid behavior during constructive schedule construction.

Deadlock is possible if the system gets locked, e.g. the situation in the case as is shown in Figure 20. To avoid this, it is required to make sure that the number of material instances residing on a subset of the resources *Rc* does not exceed some number r_c. For instance, in the case of Figure 20, the number of material instances in each of the dashed squares may not exceed 2.

b-iv When considering a constructive scheduling algorithm, a partial selection $D_{lp} \in \mathcal{D}_1$ can only be extended with a task *t'* and related definition elements to form an extended partial selection $D_{lp}^{'} \in \mathcal{D}_1$ if no WIP ceiling constraints are violated for the extended partial selection.

Let $WIPceil \in \mathcal{P}(\mathcal{P}(\mathcal{R}) \times \mathbb{N})$ be the set of applicable combinations of Rc and n_c as described earlier. Then the additional constraint is defined as follows:

$$(\forall Rc, n_c) : (Rc, n_c) \in WIPceil : (\sum r : r \in Rc : |S_m(r, tchainr(P_{lr}(D'_{lp}, r)))|) \leq n_c \qquad (2.12)$$

In many cases, material transport is performed by resources that can contain only one material instance (one-lane logistic path). This means that the only possible next transport task for such a resource is to transport the material instance further. However, when a constructive scheduling algorithm is applied this can lead to deadlock, as there might be no resource available to receive material, whereas the same resource has to make room on these resources. To avoid these kinds of deadlock situations, the scheduling algorithm has to look a bit further in life of this material instance than the next task only.

To describe these situations, the concept of tied precedences $Pt_2 \subseteq P_2$ is introduced. When choice of tasks is not considered, like in the generalized job shop scheduling problem, this implies that the subsequent transport tasks of a certain material instance that have to be executed uninterrupted after another are connected by tied precedences. A tie is defined as a chain of tasks that are connected by tied precedences. An open tie is defined as a tie of which at least one but not all tasks are selected.

When considering a constructive scheduling algorithm, the additional constraints concerning ties are as follows:

b-v A partial selection $D_{lp} \in \mathcal{D}_1$ can only be extended with a task t' from a tie to form an extended partial selection $D'_{lp} \in \mathcal{D}_1$ if it is possible to subsequently select an entire tie.

b-vi If a partial selection contains an open tie, it can only be extended with a tied task.

In the following, we formulate the possible schedule extensions for a constructive scheduling algorithm.

Let T_{lp}, I_{lp}, P_{lp} be the tasks, involved resources and precedence relation of partial selection D_{lp}. Let T_{le}, I_{le}, P_{le} be the task t', involved resources with t' and precedence edges to t' of selection extension D_{le}. Let $T'_{lp}, I'_{lp}, P'_{lp}$ be the tasks, involved resources and precedence relation of extended partial selection $\mathrm{D}'_{\mathrm{lp}}$, which is equal to $T_{lp} \cup T_{le}, I_{lp} \cup I_{le}, P_{lp} \cup P_{le}$.

Let function $Et : (\mathcal{D}_2 \times \mathcal{D}_1) \rightarrow \mathcal{P}(\mathcal{T}_1)$ be the function that determines for a partial selection D_{lp} all eligible tied next tasks, considering the tied precedence relation.

$$Et(D_2,D_{lp}) = \{t \in \mathcal{T}_2 \setminus T_{lp} | \forall t' : (t',t) \in P_2 : t' \in T_{lp})\} \tag{2.13}$$

Let function $Et_t : (\mathcal{D}_2 \times \mathcal{D}_1) \rightarrow \mathcal{P}(\mathcal{T}_1)$ be a function that determines for a partial selection D_{lp} all eligible next tasks, considering the tied precedence relation:

$$Et_t(D_2,D_{lp}) = \{t' | t' \in Et(D_2,D_{lp}) \wedge (\exists t : t \in T_{lp} : (t,t') \in P_{2t})\} \tag{2.14}$$

Let function $Er : (\mathcal{D}_2 \times \mathcal{T}_1) \rightarrow \mathcal{P}(\mathcal{R})$ be a function that determines for a task all eligible sets of involved resources, considering the available resources for the capabilities involved.

$$Er(D_2,t) = \{rr \subseteq R | (\forall r,c : r \in rr, c \in I_2(t) : r \in A(c))\} \tag{2.15}$$

Let function $check_{Cb-i}$:through $check_{Cb-iii} : (\mathcal{D}_2 \times \mathcal{D}_1 \times \mathcal{T}_1 \times \mathcal{P}(\mathcal{R})) \rightarrow \mathbb{B}$ be functions that check whether or not for a next task t' and involved resources rr' constraints b-i through b-iii hold.

$$\begin{aligned} check_{Cb-i}(D_2,D_{1p},t',rr') = (\forall t : t \in T_{1p} \wedge (t,t'0 \in P_{2m}(m) \\ : I_1(t) \cap (\cup m : m \in Ce_2(t).1 : A(Ce_2(t).0)) \\ = rr' \cap (\cup m : m \in Cb_2(t'0.1 : A(Cb_2(t').0)) \\ ) \end{aligned} \tag{2.16}$$

$$\begin{aligned} check_{Cb-ii}(D_2,D_{1p},t',rr') = (\forall m,r_b,r_e : \; m \in M_2 \wedge r_b,r_e \in R, \\ \{r_b\} = rr' \cap (\cup m : m \in Cb_2(t').1 : A(Cb_2(t').0)) \\ \{r_e\} = rr' \cap (\cup m : m \in Ce_2(t').1 : A(Ce_2(t').0)) \\ : r_b = r_e \vee (r_b,r_e) \in Mf \\ ) \end{aligned} \tag{2.17}$$

$$\begin{aligned} check_{Cb-iii}(D_2,D_{1p},T',rr') = \\ (\forall r : r \in rr' : | S_m(r, tchainr(P_{1r}(D_{1p},r)) + +t') | \leq Rm(r)) \end{aligned} \tag{2.18}$$

Let function $check_{b-iv} : (\mathcal{D}_2 \times \mathcal{D}_1 \times \mathcal{T}_1 \times \mathcal{P}(\mathcal{R}) \times \mathcal{P}(\mathcal{P}(\mathcal{R}) \times \mathbb{N}) \rightarrow \mathbb{B}$ be a function that checks whether or not for a next task t' and involved resources rr' constraint b-iv holds.

$$\begin{aligned} check_{Cb\text{-}iv}(D_2,D_{lp},t',rr,WIPceil) = \\ (\forall (Rc,n_c) \in WIPceil : (\textstyle\sum r : r' \in Rc \cap rr' : |S_m(r, tchainr(D_{lp},r) + +t')|) \leq n_c)) \end{aligned} \tag{2.19}$$

Let function $\mathrm{E} : (\mathcal{D}_2 \times \mathcal{D}_1) \rightarrow \mathcal{P}(\mathcal{T}_1 \times \mathcal{P}(\mathcal{R}) \times \mathcal{P}(\mathcal{T}_1 \times \mathcal{T}_1))$ be the function that returns for an unselected TRS definition all possible extensions e in the form of task t', involved resources rr' and precedences pr' with which partial schedule D_{lp} can be extended to form an extended partial schedule D'_{lp}. D_{le} can be determined from e by taking $e.0$ for T_{le}, ($e.0$, $e.1$) for I_{1e} and $e.2$ for P_{le}. Then, function E can be defined as follows:

$$\begin{aligned}
&E(D_2 D_{1p}) = \\
&\quad \{ (t' \\
&\quad\quad ,rr' \\
&\quad\quad ,\{(t,t') | (t,t') \in P_2 \\
&\quad\quad \vee ((I_{1p}(t) \cap rr' \neq \emptyset) \wedge (\nexists t'' \in \mathrm{T}_{1t} : \mathrm{I}_{1p}(t'') \neq \emptyset \wedge (t,t'') \notin P_{1p})) \\
&\quad\quad \} \\
&\quad\quad) \\
&\quad | \; Et_t(D_{1p}) = \emptyset \Rightarrow t' \in Et(D_2, D_{1p}) \wedge Et_t(D_{1p}) \neq \emptyset \Rightarrow t' \in Et_t(D_2, D_{1p}) \\
&\quad \wedge rr' \in Er(D_2, t') \\
&\quad \wedge check_{b\text{-}i}(D_2, D_{1p}, t', rr') \\
&\quad \wedge check_{b\text{-}ii}(D_2, D_{1p}, t', rr') \\
&\quad \wedge check_{b\text{-}iii}(D_2, D_{1p}, t', rr') \\
&\quad \wedge check_{b\text{-}iv}(D_2, D_{1p}, WIPceil, t', rr') \\
&\quad \wedge \exists D'_{1e} \in D_1 : Et_t(D'_{1p} \cup D'_{1e}) = \emptyset \\
&\quad \}
\end{aligned} \tag{2.20}$$

In function E, both the constraints involved in generalized job shop scheduling, and the additional machine-specific scheduling constraints b-i though b-vi can be recognized.

First, additional machine-specific constraints are introduced to avoid unfeasible timing behavior, then the transformation function is described.

To describe hardware interference, some additional definition elements are defined. Subsequently, constraints are defined using these additional elements. Furthermore, the constraint that must be satisfied in order to time a selected TRS is described. Finally, constraints with respect to task start and finish time are described.

Some state transitions of some resources can only take place synchronously with state transitions of other resources.

$Ts \subseteq \mathcal{P}(\mathcal{R} \times \mathcal{S} \times \mathcal{S})$ gives the subsets of synchronous resource state transitions.

In a machine, certain areas exist in which resources can collide. These areas are allowed to be visited by the resources mutual exclusively. This additional constraint is described by adding a resource to $\mathcal{R}$ for such a hazardous areas, and involve this resource in every resource state transition that visits this area as described above. For the collision area resources, physical states do not play a role.

$\mathcal{R}_c$ is a finite set whose elements are called collision areas.

$Tc : \mathcal{R} \times \mathcal{S} \times \mathcal{S} \rightarrow \mathcal{R}_c \cup \{\emptyset\}$ gives the collision area resource that is associated with a certain resource state transition.

The setup transitions might consist of several elementary subtransitions, each of which might be involved with forced synchronism or collision avoidance.

$Te : \mathcal{R} \times \mathcal{S} \times \mathcal{S} \rightarrow (\mathcal{R} \times \mathcal{S} \times \mathcal{S})*$ gives the elementary subtransitions of a resource state transition. In case there are no elementary subtransitions, the original state transition is returned.

Constraints:

i Every task is elementary:

$$(\forall t, r : t \in \mathcal{T}_1 \wedge r \in I_1(t) \cap (\mathcal{R} \setminus \mathcal{R}_c) : Te(r, Sb_1(t,r), Se_1(t,r)) = [(r, Sb_1(t,r), Se_1(t,r))])$$

ii Every task matches *Ts*, which implies that for each task *t* either the resource state transitions involved encapsulate some set of synchronous state transitions *s* from *Ts*, or none of the involved resource state transitions occurs in any *s* from *Ts*.

$$(\forall t : t \in \mathcal{T}_1 : (\exists s \in Ts : s \subseteq (\cup r : r \in I_l(t) : \{(r, Sb_l(t,r), Se_l(t,r))\})) \vee (\forall s : s \in Ts : s \cap (\cup r : r \in I_1(t) : \{(r, Sb_l(t,r), Se_l(t,r))\}) = \emptyset))$$

iii Every task matches *Tc*, which implies that for each task and each involved resource (excluding hazardous areas) goes that for each resource state transition any involved collision area is involved in the task too:

$$(\forall t, r : t \in \mathcal{T}_1 \wedge r \in I_l(t) \cap (\mathcal{R} \setminus \mathcal{R}_c) : Tc(r, Sb_1(t,r), Se_l(t,r)) \in I_l(t))$$

Note that these constraints essentially hold for every TRS definition level (main = level 1).

1-t A selected TRS $D_1 \in \mathcal{D}_1$ is timeable if subsequent task end and begin states in the chain of tasks per resource match:

$$(\forall t, t', r : (t,t') \in tchainr(P_{1r}(D_1, r)) \wedge rI_1(t(\cap \in I_1(t') \cap (\mathcal{R} \setminus \mathcal{R}_c) : Se_1(t,r) = Sb_1(t',r))$$

Note that although a number of possible sequences of elementary resource state transitions between some possible resource state transition that is implied by selection may be allowed, only one transition is defined by function *Te*. Furthermore, for each elementary state transition at most one collision area is defined by function *Te*.

When taking the issue of forced synchronous and elementary state transitions into account, it is possible that a setup state transition of a resource implies setup state transitions of other resources. These implied state transitions also have to match the states of the resource in turn, which might imply other state transitions, etcetera. To avoid an infinite chain reaction caused by loops, additional constraints are defined.

A core setup transition is defined as the resource state transition of a resource r from the end state of the previous task on *r* to the begin state of task $t \in \mathcal{T}_2$, in case these states do not match. Using this definition, the following constraints have to be satisfied to prevent loops during the transformation into a timeable TRS:

- For two core setup transitions necessary for one task *t*, the sets of resources involved in setup transitions implied by each core setup transition do not overlap. For example, when there are two core setups for resource A and B, and the core setup for resource A implies a synchronous state transition of resource C, then the state transitions implied by the core setup for resource B may not involve resource C.
- For any setup transition of resource *r'* (either core or implied by other setup transitions), the set of resources involved in setup transitions implied by this setup transition does not contain *r'* itself. For example, it is not allowed that a state transition of resource B that is implied by a state transition of resource A on its turn implies a state transition of resource A.

a-i To transform a timeable selected TRS D_I to a timed TRS D_0, besides the constraints presented in the definition of the general problem, additional constraints can be introduced for the time between certain task start or end times. Examples are the post exposure bake time and the time that a substrate resides at a load robot.

Due to the constraints to ensure one possible finite transformation A of a selected TRS D_I to a timeable selected TRS D_1^T, this transformation can be defined by a function. In the chain of tasks per resource of a definition D_I, additional setup tasks are introduced such that a chain of tasks results that satisfies constraints i, ii, iii, and 1-t to result in a timeable selected TRS D_1^T.

Let *insert* be a function inserting setup tasks and precedence edges in a selected TRS D_1 for all non-matching subsequent task end and begin states defined by $insert(D_1) = D_1^{'}$ such that:

$$\begin{array}{ll}
(\forall r,t,t' & : r \in \mathcal{R} \wedge t,t' \in T_1 : (t,t') \in P_{1r}(D_1,r) \wedge Se_1(r,t) \neq Sb_1(r,t') \\
 & (\exists t'' : t'' \in \mathcal{T}_1^{'} \wedge (t,t') \notin P_1^{'} \wedge (t,t'') \in P_1^{'} \wedge (t'',t') \in P_1^{'} \\
 & : I_1^{'}(t'') = \{r\} \wedge Sb_1^{'}(t'',r) = Se_1(t,r) \wedge Se'_1(t'',r) = Sb_1(t',r) \\
 &) \\
) &
\end{array} \qquad (2.21)$$

and $\mathcal{T}_1^{'}, P_1^{'}, I_1^{'}, Sb_1^{'}, and\ Se_1^{'}$ are minimal.

Let $e = Te(r, Sb_1(t,r), Se_1(t,r))$. Let *decomp* be a function decomposing any compound transitions in setup tasks of a selected TRS to match Te defined by $decomp(D_1) = D_1^{'}$ such that:

$$\begin{array}{ll}
(\forall t,r & : t \in \mathcal{T}_1, r \in I_1(t) \wedge Te(r, Sb_1(r,t'), Se_1(r,t)) \neq (r, Sb_1(r,t), Se_1(r,t)) : \\
 & (\forall 0 \le i < len(e) : \\
 & (\exists t' : t' \in T_1^{'} : I_1^{'}(t') = \{r\} \wedge \mathrm{Sb}_1^{'}(t',r) = e.i.1 \in \wedge \mathrm{Se}_1^{'}(t',r) = e.i.2 \\
 & \wedge i = 0 \Rightarrow (\forall t'' : t'' \in T_1 \wedge (t'',t) \in P_{1\mathrm{r}}(D_1,r) : (t'',t) \notin P_1^{'} \wedge (t',t'') \in P_1^{'} \\
 & \wedge i = len(e) - 1 \Rightarrow (\forall t'' : t'' \in \mathcal{T}_1 \wedge (t,t'') \in P_{1r}(D_1,r) : (t,t'') \notin P_1^{'} \wedge (t',t'') \in P_1^{'}) \\
 &) \\
 &) \\
) &
\end{array} \qquad (2.22)$$

and $\mathcal{T}_1^{'}, P_1^{'}, I_1^{'}, Sb_1^{'}, and\ Se_1^{'}$ are minimal.

Let *notsync:* $D_1 \rightarrow T_1$ be a function determining which tasks of a selected TRS are not matching *Ts*

$$notsync(\mathrm{D}_1) = \{t \in T_1 | \exists s : s \in Ts : s \not\subseteq \{(r, Sb_1(t,r), Se_1(t,r)) | r \in I_1(t)\}\} \qquad (2.23)$$

Let *addsync* be a function adding forced synchronous resource state transitions to setup tasks of a selected TRS which are not matching Ts defined by $addsync(D_1) = D_1^{'}$ such that:

$$\begin{array}{l}(\forall t : t \in notsync(D_1) :\\ (\forall s, ts : s \in Ts, ts \in s\\ : ts \cap \{(r, Sb_1(t,r), Se_1(t,r)) | r \in I_1(t)\} \neq \emptyset \Rightarrow ts.0 \in I_1'(t) \wedge ts = \{(r, Sb_1'(t,r), Se_1'(t,r))\}\\ )\\ )\end{array} \quad (2.24)$$

Let *addcoll* be a function adding collision areas to setup tasks of a selected TRS which are not according *Tc:* $addcoll(D_l) = D_l'$ such that:

$$(\forall t, r : t \in \mathcal{T}_1 \wedge r \in I_1(t) \cap (\mathcal{R} \setminus \mathcal{R}_c) \wedge Tc(r, Sb_l(t,r), Se_l(t,r)) \neq \emptyset : Tc(r, Sb_l(t,r), Se_l(t,r)) \in I_l'(t)) \quad (2.25)$$

To transform a selected TRS $D_l|$ that does not satisfy constraint 1-t into a selected D_1^T that does satisfy constraint 1-t, function $trans_{A-t} : \mathcal{D}_1 \rightarrow \mathcal{D}_1'$ is defined as follows:

$$trans_{A-t}(D_1) = \begin{cases} addcoll(decomp(insert(D_1))) & \text{if } notsync(decomp(insert(D_1))) = \emptyset \\ trans_{A-t}(addsync(decomp(insert(D_1)))) & \text{if } notsync(decomp(insert(D_1))) \neq \emptyset \end{cases} \quad (2.26)$$

Note that $trans_{A-t}$ can also be used in a constructive algorithm.

The algorithm to optimize timing of a timeable selected TRS D_1^T given the timing constraints a is a linear programming problem that is solvable by well known techniques. Linear programming can also be used in a constructive algorithm.

For an example system, the additional elements can be defined as follows:

$\mathcal{M} = \{W1, W0, ...\}$.

$Cb_2 = \{(W1 - T2L, \{(T, \{W1\}), (L, \{\})\}), (W1 - L2R, \{(L, \{W1\}), (R, \{\})\}), ...\}$.

$Ce_2 = \{(W1 - T2L, \{(T, \{\}), L, \{W1\})\}), ...\}$

$Rm = \{(T0, 1), (L0, 1), ...\}$

$Mf = \{(T0, L0), (L0, R0), (R0, A0)$

$$\begin{array}{l}Pt_2 = \{(\mathrm{W1-T2L, W1-L2R}), (\mathrm{W1-L2R, W1-R2A}), (\mathrm{W1-R2A, W1-AL}),\\ (\mathrm{W1-AL, W1-A2R}), (\mathrm{W1-A2R, W1\text{-}R2S}), (\mathrm{W1\text{-}S2R, W1\text{-}R2L}),\\ (\mathrm{W1\text{-}R2L, W1\text{-}L2T}), ...\}\end{array}$$

$WIPceil = \{(\{L0, R0, A0, S0\}, 2), (\{L1, R1, A1, S1\}, 2)\}$

$Ts = \{\{(S0, @lm, @e), (S1, @e, @lm)\}, \{(S1, @lm, @e), (S0, @e, @lm)\}\}$

$Rc = \{HA\}$

Tc={((*R0*,@*ca*,@*l*),*HA*), ...}

Te={((*R0*,@*1*,@*a*),[(*R0*,@*1*,@*ca*),),[(*R0*,@*ca*,@*ca*)]),((*S0*,@*lm*,@*u*), [(*S0*,@*lm*,@*e*),(*R0*,@*e*,@*u*)]), ...}

Using a simple heuristic, namely fill up the system as much as possible and 'Earliest Start First', a feasible and valid schedule is obtained. Subsequently, a timing post-processing step is done to fulfill the additional time window constraints $\tau_{F_0}(W1-R2S)-\tau_{s_0}(W1-A2R)=3[s]$, etc., and $\tau_{F_0}(W1-L2T)-\tau_{s_0}(W1-EXP)=50[s]$, etc..

After this, the schedule of Figure 21 is obtained. It can be concluded that the schedule matches all additional restrictions.

The critical path as is displayed in Figure 22 is as desired: steady-state exposure is on the critical path.

A third embodiment of the invention is a lithographic apparatus using extreme ultraviolet radiation as the exposure radiation connected to a track. The substrate flow of the third embodiment is schematically shown in Figure 23. The circles in the figure represent the resources of the model – one track T, four load locks L0-L3, two robots R0, R1 and two substrate tables WT0, WT1 (also referred to as chucks). The arrows in the figure represent the possible substrate flows through the apparatus. A fresh substrate starts in the track, it is transported to one of the tables, where it is measured and exposed and finally it is transported back to the track.

The track delivers fresh substrates to the lithographic apparatus and retrieves exposed substrates from the lithographic apparatus. A simple model of the track T is used for the purposes of this description, where each time a substrate is retrieved or a new substrate must be delivered, the track needs a certain amount of time to perform internal actions.

The load locks L0-L3 have room for one substrate and are bi-directional, which means that a single load lock can be used for both ingoing and outgoing substrates. After a substrate is put in a load lock, the pressure in the load lock is brought to vacuum for an ingoing substrate or to atmospheric for an outgoing substrate.

The robots R0, R1 in the substrate handler both have two arms, placed at 180° opposite of each other. Each arm can carry one substrate. The robots rotate their arms between the load locks and the chucks. A robot arm can reach two of the four locks when it is at the lock side and it can reach both chucks when it is at the chuck side.

The chucks or substrate tables WT0, WT1 each have room for one substrate. The substrate tables may adopt positions – the load/unload position, the measure position and the exposure position. However, it is also possible to combine the load/unload position with the measure position so that the substrate tables have only two positions. When the substrate table is at measure position the substrate can be loaded or unloaded by one of the robots. Measurement also requires the substrate table in this position. Exposure takes place at the exposure position. Both substrate tables change positions synchronously during the so-called 'chuck swap'.

As with the first and second embodiments, the system can be described as an instantiated, unselected TRS definition (class 2). The control strategy of the third embodiment is embodied in a heuristic filter configuration.

In this embodiment, there is one track and there are four load locks, two robots and two substrate tables, which would mean nine resources. However, instead of modeling the robots with a substrate capacity of two substrates, the robot arms of the robots are defined as separate resources that represent separate and unique substrate locations. This is done so because the location of a substrate must be known for the robot, as it influences the behavior concerning the robot rotations. When the robot is modeled as one resource with a material capacity of two substrates, it is unknown on which arm a substrate is residing. This means there are eleven resources, which are numbered from 0 through 10 and stored in $\mathcal{R}$.

It is clear that there are four capabilities in the model, namely track, lock, robot arm and substrate table. These capabilities are numbered from 0 through 3 and stored in $\mathcal{C}$.

Several behaviors can be performed by the system. Most of the behaviors concern substrate transport between two resources. The strings representing these transport behaviors are of the form "capability2capability". For example the transport from the track (T) to a load lock (L) is called "T2L". There are six transport behaviors, namely "T2L", "L2R", "R2C", "C2R", "R2L", and "L2T". Besides these transport behaviors, there are two other behaviors, "measure" and "expose". All eight behaviors involve certain capabilities, which is defined in I_2. It is clear that for the transport behaviors, the involved capabilities are those where the material is transported between. For the "measure" and "expose" behaviors the involved capability is a substrate table.

In *A*, the available resources per capability are defined, which is straightforward. All resources except the track have a material capacity of one substrate, which is defined in *Rm*. The track in the model has infinite room for substrates, but for convenience the material capacity of the track is set to 100 substrates. The possible material flow between resources is defined in *Mf*. From the track, substrates can be transported to each of the four load locks and from each load lock, substrates can transported to two of the four robot arms. Which robot arms those are, depends on the load lock number corresponding to the arrows in Figure 23. From the robot arms, substrates can be transported to each substrate table. In the substrate transport from the substrate table back to the track, the reversed material flow as described above is possible. The last two definition elements of the static system definition, *Te*, and *Ts*, concern the resource state transitions. Before these are explained, the possible states and state transitions of each resource are defined.

In the track, there is one possible state transition that consists of elementary subtransitions, namely from 'retrieved exposed substrate' to 'ready to retrieve'. This state transition is needed when the track retrieves two exposed substrates subsequently (which will happen at the end of the schedule). This transition consists of two elementary state transitions, namely, the track internal action from 'retrieved exposed substrate' to 'ready to send' (see Figure 24) and the bypass transition from 'ready to send' to 'ready to retrieve'. This breakdown into elementary subtransitions is defined in *Te*. For the locks, the state denotes whether the air is at atmospheric pressure or at vacuum.

The transport tasks involving a lock do not change the state of the lock. Pumping down to vacuum and venting of air (which are both resource setups) are the possible state transitions that change the state. The automaton of a lock is shown in Figure 25.

The robot arms and the substrate tables both have two possible states with transitions between them. The robot arms have as state their position, either at lock side or at substrate table side. The state of a substrate table tells whether it is at measure position or at exposure position. The robots and the substrate tables both have forced synchronism, denoted with *Ts* in the automata of Figures 26 and 27 (which also shows the tasks that do not change the resource state). When a robot rotates, the robot arms switch positions, which means that the state transitions of both arms are performed synchronously. The same holds for the substrate tables during the chuck swap, when one substrate table changes position, the other substrate table also changes position.

These synchronous state transitions of the robot arms and of the substrate tables are defined in *Ts*.

Besides the static system definition described above, other static information must also be defined. The zero order durations for each behavior and for all elementary resource state transitions are defined in *Dtz* and *Dsz*, respectively. Because no higher order duration functions are defined which require the hardware capacities defined in *Hc* (like acceleration and maximum velocity), it is not necessary to define these hardware capacities. Finally, the resource descriptions are defined in *resdescr*.

The work configuration contains all dynamic information of the TRS, consisting of the dynamic system definition and the initial situation.

The dynamic system definition consists of the definition elements $\mathcal{T}_2$, $\mathcal{L}_2$, $\mathcal{G}_2$, Ln_2, Gn_2, Ga_2, P_2, Pt_2, Tb_2, Sb_2, Se_2, Cb_2, and Ce_2.

In the third embodiment, there are no alternatives concerning tasks, so only tasks are defined and no clusters or groups. For each substrate, eight tasks are defined which correspond to the behaviors mentioned in the previous section. To get a correct manufacturing process for each substrate, the essential precedence relations are defined in P2 between the eight tasks for one substrate, which results in the following life of a substrate: "T2L", "L2R", "R2C", "measure", "expose", "C2R", "R2L", and finally "L2T". Also the order of substrates going in the lithographic apparatus is defined using precedence relations between the ingoing tasks (in this case "T2L") of the different lives of a substrate, resulting in an inflow order with increasing substrate numbers. None of these precedence relations is tied, so Pt_2 is empty.

The behaviors mentioned above are assigned to each task in Tb_2. The corresponding task graph that shows three lives of a substrate is shown in Figure 28.

In *Sb2* and *Se2*, the begin and the end states of each involved capability are defined for each task. In the third embodiment, all transport tasks require the involved capabilities to be at the correct position, such that the transport can be performed. Their state (which is their position) remains unchanged during the task, so the end state is equal to the begin state (see also the automata in Figures 24 through 27). However, the state of the track does change during transport tasks. For delivery of a fresh substrate (behavior "T2L"), the track needs to be in state 'ready to send', while at the end of the task, the track state is 'ready to retrieve'. For the retrieval of exposed substrates (behavior "L2T"), the required begin state is 'ready to retrieve' and the end state is 'retrieved exposed substrate'. For "measure" and "expose" tasks, the begin state and the end state are 'at measure position' and 'at exposure position', respectively.

Finally, the begin and the end material configurations are defined in Cb_2 and Ce_2, respectively. Obviously, for a transport, the delivering capability holds the material at the begin of the task and the receiving capability holds the material at the end of the task. For "measure" and "expose" tasks, the material stays on the capability (in this case a substrate table), so Cb2 and Ce2 are equal for these tasks.

Besides the dynamic system definition described above, the initial situation must be defined. The initial heap is empty, as no tasks are performed yet. For each resource, the initial time (defined in the initial contour) is zero. The initial contour also contains all initial resource states, which is 'ready to send' for the track and 'at atmospheric pressure' for the load lock. For the robot arms and substrate tables, the initial state is one of the two possible positions, whereas the opposite arm or the other substrate table has the other possible position as initial state.

Next, the heuristic filter configuration design for substrate flow in the third embodiment is described.

Material flow with logistic crossings in a TRS may lead to deadlock situations, which is the case in the present embodiment. After examination of the resource configuration and the possible material flows, the following possible deadlock situations are recognized. First, in a combination of two load locks and a robot, it is possible that both locks have fresh substrates in them and that both robot arms hold an exposed substrate. In this case there is no robot arm free for the fresh substrates and there is no empty lock for the exposed substrates, which means that no further tasks can be performed. An example of this type of deadlock is depicted in Figure 29, where an arrow denotes a substrate with the direction of the next transport task in its material life.

Also behind the locks, between the robots and substrate tables, a deadlock situation may occur. This is the case when all four robot arms have fresh substrates and both substrate tables are occupied by exposed substrates. This situation is shown in Figure 30.

Because deadlocks are examples of invalid behavior, constraining filters are used to prevent them. In this case, *WIP ceiling* is used. For each of the three possible deadlock situations as described above (two lock-robot combinations and one behind the locks), a *maxwip* filter is defined that checks the *WIP ceiling* constraint for that group of resources. In the implementation of the T-ReCS scheduler, these three instances of the *maxwip* filter are combined into *one maxwip* filter with the three *WIP ceiling* constraints as filter parameters.

For both lock-robot combinations (two locks and two robot arms), a *WIP ceiling* of three substrates is set in the filter parameters. The parameters of these *maxwip* filters are the resource numbers of the two locks and the two robot arms, together with the maximum WIP of three substrates. With this heuristic filter, the type of deadlock depicted in Figure 29 is prevented. The third *maxwip* filter concerns all resources behind the locks (four robot arms and two substrate tables) and has as parameters their resource numbers and a maximum WIP of five substrates. With this filter, the situation shown in Figure 30 cannot be reached any more.

After analysis of the deadlock situation in the counterexample, it became clear that lowering the maximum WIP level behind the locks to four substrates would prevent this deadlock situation. Besides this situation, a similar deadlock situation as in Figure 31 can still be reached when eight substrates are in the lithographic apparatus. When all locks and one robot arm of each robot hold a fresh substrate, while each substrate table holds an exposed substrate, no WIP ceiling constraint (including the one lowered to four substrates) is violated, however no further tasks can be performed. This situation is depicted in Figure 32. Again note that this is a deadlock only for the modeled TRS and the used filters. To prevent the last deadlock situation with eight substrates, a fourth *maxwip* filter is added. This filter has the resource numbers of all resources except the track and a maximum WIP of seven substrates as parameters.

This table shows the initial and redesigned heuristic filter configuration. The design process described above only concerns the constraining filters, in this case resulting in four maxwip filters. All generated schedules within this heuristic filter con- figuration will have valid behavior, which in this case means that the verified property, deadlock freeness, is always satisfied. However, within the valid behavior, some behaviors are more preferred than others. Therefore, two comparing filters are added to the heuristic filter configuration.

To get high resource utilization, it is desired that the number of substrates in the lithographic apparatus is as high as possible. Therefore, the *fillmaxwip* filter is added to the heuristic filter configuration. The parameters of this filter are the resources where the WIP level should be as high as possible, so in this case all the resources except the track.

Finally, the total schedule time should be as low as possible. This can be accomplished by preferring tasks which have the lowest start times. Therefore, the earliest start first (ESF) filter is used.

Concluding, the heuristic filter configuration consists of four instantiations of the constraining *maxwip* filter (implemented in one filter), and two comparing filters, the fillmaxwip and the earliest start first (ESF) filter.

The resulting Gantt chart is shown in Figure 33 and the critical path in Figure 34. The lives of all substrates are nicely interweaved, and all exposure tasks are on the critical path. Critical exposures are also desired behavior, because the lens involved in this task is the most expensive part of the lithographic apparatus and should have maximum utilization. As all exposures are on the critical path, the lens has the highest utilization possible, which means that this is an optimal schedule concerning exposures.

However, looking at the inflow and the outflow of the substrates (see the substrate numbers in the figure), the order is not FIFO.

The next experiment demonstrates that it is possible to generate schedules with FIFO order concerning the inflow and the outflow of substrates. The scheduler is guided towards this behavior by applying an additional heuristic filter which makes sure that substrates cannot overtake each other. This filter is the *incrmatnr* filter that chooses the task with lowest material number when more tasks with the same behavior for different materials are eligible. This filter can be used, because all substrates follow the same logistic path and they enter the lithographic apparatus with increasing substrate number. The only filter parameter of the *incrmatnr* filter is a list containing the behaviors to which the filter must be applied. When this filter is applied to all behaviors, it is guaranteed that substrates cannot overtake each other and will come out of the system with increasing substrate numbers, which means FIFO order. When this heuristic filter configuration is used in the scheduler, the generated heap (using one shot scheduling) is indeed FIFO. The Gantt chart of this schedule with the substrate numbers is shown in Figure 35. The schedule is still optimal concerning exposures (which are all on the critical path) and the total schedule time is not increased compared to the schedule found earlier, which means that this schedule is better.

The final experiment concerns the flattening of the post expose bake (PEB) time variability for embodiment. The timing post-processor tool is applied and after fine-tuning of the weight functions, the PEB time machine flow variability is reduced to zero. The Gantt chart of the FIFO schedule with flattened PEB time variability is shown in Figure 36, where the second exposure (on substrate table 0) is delayed (resulting in a gap in the schedule). Figure 37 shows the PEB times for each substrate for the non-FIFO schedule (I), the FIFO schedule without PEB flattening (II), and the FIFO schedule with PEB flattening (III). The FIFO schedule shows a reasonable improvement in PEB time variability compared to the non-FIFO schedule, however it is not zero, which is accomplished by using timing post-processing.

While specific embodiments of the invention have been described above, it will be appreciated that the invention may be practiced otherwise than as described. The description is not intended to limit the invention.